

TADM51

Database Administration Oracle

SAP NetWeaver

Date _____
Training Center _____
Instructors _____
Education Website _____

Participant Handbook

Course Version: 62
Course Duration: 5 Day(s)
Material Number: 50089428



An SAP course - use it to learn, reference it for work

Copyright

Copyright © 2008 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Trademarks

- Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.
- IBM®, DB2®, OS/2®, DB2/6000®, Parallel Sysplex®, MVS/ESA®, RS/6000®, AIX®, S/390®, AS/400®, OS/390®, and OS/400® are registered trademarks of IBM Corporation.
- ORACLE® is a registered trademark of ORACLE Corporation.
- INFORMIX®-OnLine for SAP and INFORMIX® Dynamic Server™ are registered trademarks of Informix Software Incorporated.
- UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.
- Citrix®, the Citrix logo, ICA®, Program Neighborhood®, MetaFrame®, WinFrame®, VideoFrame®, MultiWin® and other Citrix product names referenced herein are trademarks of Citrix Systems, Inc.
- HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
- JAVA® is a registered trademark of Sun Microsystems, Inc.
- JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
- SAP, SAP Logo, R/2, RIVA, R/3, SAP ArchiveLink, SAP Business Workflow, WebFlow, SAP EarlyWatch, BAPI, SAPPHIRE, Management Cockpit, mySAP.com Logo and mySAP.com are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other products mentioned are trademarks or registered trademarks of their respective companies.

Disclaimer

THESE MATERIALS ARE PROVIDED BY SAP ON AN "AS IS" BASIS, AND SAP EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THESE MATERIALS AND THE SERVICE, INFORMATION, TEXT, GRAPHICS, LINKS, OR ANY OTHER MATERIALS AND PRODUCTS CONTAINED HEREIN. IN NO EVENT SHALL SAP BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR PUNITIVE DAMAGES OF ANY KIND WHATSOEVER, INCLUDING WITHOUT LIMITATION LOST REVENUES OR LOST PROFITS, WHICH MAY RESULT FROM THE USE OF THESE MATERIALS OR INCLUDED SOFTWARE COMPONENTS.

About This Handbook

This handbook is intended to complement the instructor-led presentation of this course, and serve as a source of reference. It is not suitable for self-study.




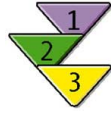

Typographic Conventions

American English is the standard used in this handbook. The following typographic conventions are also used.

Type Style	Description
<i>Example text</i>	Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths, and options. Also used for cross-references to other documentation both internal (in this documentation) and external (in other locations, such as SAPNet).
Example text	Emphasized words or phrases in body text, titles of graphics, and tables
EXAMPLE TEXT	Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example SELECT and INCLUDE.
Example text	Screen output. This includes file and directory names and their paths, messages, names of variables and parameters, and passages of the source text of a program.
Example text	Exact user entry. These are words and characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.

Icons in Body Text

The following icons are used in this handbook.

Icon	Meaning
	For more information, tips, or background
	Note or further explanation of previous point
	Exception or caution
	Procedures
	Indicates that the item is displayed in the instructor's presentation.

Contents

Course Overview	vii
Course Goals	viii
Course Objectives	viii
Unit 1: Database Overview	1
Database Architecture	3
Connecting to the database	31
Database Administration Tools.....	52
Administration of Oracle Instances	83
Unit 2: Backup, Restore & Recovery.....	107
Backup Strategy	109
Backup Tools	126
Appendix: Tape Management with BR*Tools	153
Performing Backups	171
Restore and Recovery	192
Advanced Backup Techniques	220
Unit 3: Monitors and Tools	237
Introduction to Oracle Data Management	238
Database System Check	260
CCMS Alert Monitor	288
Unit 4: Storage Management	295
Tablespace Administration.....	296
Reorganization of Tables	320
Housekeeping and Troubleshooting	343
Unit 5: Introduction to Oracle cache management	363
Oracle System Global Area	364
Oracle Program Global Area	386
Unit 6: Monitoring of the database instance	403
Monitoring Performance Using the DBA Cockpit.....	404

Unit 7: Appendix: Monitoring of the database instance.....	435
Oracle Wait Interface	436
Automatic Workload Repository and Histories.....	462
Unit 8: Analyzing application design	481
Consequences of Expensive SQL Statements	483
Using SM50/SM66 to Find Expensive SQL Statements.....	487
Using ST03/STAD to Find Expensive SQL Statements	490
Using ST04 to Find Expensive SQL Statements	494
Using the SQL Trace to Find Expensive SQL Statements	508
Exclusive Lock Waits	516
Unit 9: Index management and optimization.....	533
Index Utilization	534
Creating an Index.....	551
Unit 10: Cost Based Optimizer	575
Update Statistics.....	576
Appendix: Cost Evaluation	596
Unit 11: Analyzing physical and logical layout	615
Fragmented Indexes	616
I/O Contention.....	637
Unit 12: Analyzing memory configuration	645
Data Buffer Utilization	646
Unit 13: Appendix: Analyzing memory configuration	663
Efficiency of the Shared Pool	664
Monitoring the Automatic Program Global Area	672
Index	689

Course Overview

This course will introduce Oracle database administration on SAP systems using tools and transactions provided by SAP.

This course is based on:

- Oracle 10.2
- SAP NetWeaver AS 7.00
- BR*Tools 7.00

Course TADM51 completes your training to become a certified SAP Certified Technology Associate - System Administration - SAP NetWeaver 7.0 (Oracle).

To pass the exam to become a certified **SAP Certified Technology Associate - System Administration - SAP NetWeaver 7.0 (Oracle)**, you must have good knowledge of the content of courses TADM10 and TADM12, as well as of the database-specific course TADM51 (Oracle).

Like the other TADM courses, TADM51 comprises several individual courses (or parts thereof), which are arranged here in a way that will enable you to gain the knowledge you require as an SAP Technology Consultant as efficiently as possible.

Course TADM51 is based on content taken from the following courses:

1. ADM505 - Database Administration (Oracle) I
2. ADM506 - Database Administration (Oracle) II

At the end of this training there is a three-hour certification exam that covers topics from courses TADM10, TADM12, and TADM51.



Caution: Note that the certification exam has been designed in such a way that the answers to all of the exam questions are contained in the folders provided for courses TADM10, TADM12, and TADM51.

Therefore, you do not require any additional course material even if the instructor hands out other books during the course or provides additional information not contained in the course folder.

Target Audience

This course is intended for the following audiences:

- SAP Technology consultants (Associate Level)

Course Prerequisites

Required Knowledge

- Basic understanding of an Oracle database
- Basic knowledge of system administration of at least one operating system

Recommended Knowledge

- To pass the exam to become a certified **SAP Certified Technology Associate - System Administration - SAP NetWeaver 7.0 (Oracle)**, you should have the knowledge and skills gained by attending courses TADM10 and TADM12.



Course Goals

This course will prepare you to:

- Understand the structure of an Oracle database
- Use database management tools provided by SAP
- Develop suitable backup strategies
- Perform backups, restore, and recovery with tools provided by SAP
- Perform space and segment management with tools provided by SAP
- Understand Oracle cache Management
- Analyze performance problems



Course Objectives

After completing this course, you will be able to:

- Describe the architecture of an Oracle database
- Use database management tools provided by SAP
- Define a backup strategy
- Perform backups with use of SAP database management tools
- Perform a restore/recovery of the database using SAP tools
- Explain the concepts of Oracle data storage management
- Use SAP tools to run regular database checks
- Solve storage-related problems
- Define a strategy for refreshing the statistics used by the cost-based optimizer
- Describe Oracle cache management
- Use the SAP database monitor to analyze performance problems

- Detect expensive SQL statements
- Describe how incorrect indexes affect performance
- Analyze memory configuration of the Oracle database
- Analyze physical and logical layout of the database

Unit 1

Database Overview

Unit Overview

This unit will give you a general overview of:

- Database architecture
- How SAP and the database administration tools connect to the database
- Database administration tools
- Administration of an Oracle instance



Unit Objectives

After completing this unit, you will be able to:

- Describe the architecture and the main components of an Oracle database
- Explain the basic concepts of the Oracle database
- Identify the file structure of an Oracle database in a SAP system
- Name the default operating system and database users
- Explain Oracle communication over a network (NETServices)
- Describe the function of the Oracle listener
- Start and stop the Oracle listener
- Identify the main tools for administration of the Oracle database
- Describe basic functions of SQL*Plus
- Use SAP tools for administration of the Oracle database: BR*Tools
- Explain the usage of the SAP CCMS for the administration of Oracle databases
- Change parameters for initialization
- Start and stop the Oracle instance
- Identify and monitor diagnosis files

Unit Contents

Lesson: Database Architecture	3
-------------------------------------	---

Exercise 1: Oracle Environment Variables	27
Lesson: Connecting to the database	31
Exercise 2: Establish connection to the database	47
Lesson: Database Administration Tools	52
Exercise 3: Database Administration Tools	79
Lesson: Administration of Oracle Instances.....	83
Exercise 4: Change Oracle Parameters.....	93

Lesson: Database Architecture

Lesson Overview

This lesson explains Oracle architecture. To understand administration, and how restore and recovery works, a general understanding of the Oracle architecture is necessary.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe the architecture and the main components of an Oracle database
- Explain the basic concepts of the Oracle database
- Identify the file structure of an Oracle database in a SAP system

Business Example

You want to install an SAP system with an Oracle database in your company. Before the installation, the disk layout on the server has to be planned and prepared.

There are general recommendations in the installation guide for the disk layout, but in practice, the distribution of database components on different disks is a compromise between security, optimal disk usage, and performance.

You need the background to design a good disk layout for your purpose.

Introduction

An Oracle server is a relational database management system (RDBMS), that is, a server component that manages a relational database model.

An RDBMS (or a database server) is able to:

- Manage large amounts of data in a multi-user environment so that many users can concurrently access the same data
- Maintain relationships between data
- Provide secure access to data using over a user authorization concept
- Recover data automatically to the most recent consistent status after a system failure
- Deliver high performance for processing data requests

In an SAP system, the only interactive user connecting to the database server should be the database administrator. Application data processing is almost exclusively performed by work processes of SAP instances in the role of database clients.

Terminology

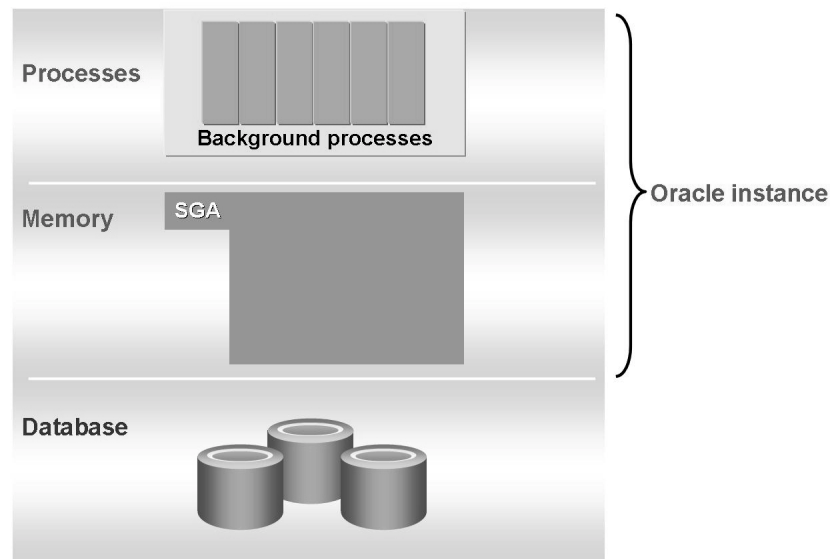


Figure 1: Database Terminology

database

An Oracle database is a collection of data, logically treated as a unit. The data is physically stored in one or several files. Oracle manages data in logical units called **tablespaces**. A database object, such as a table, is always created in a particular tablespace. A tablespace consists of one or more files.

instance

As the database is only a passive part of a database server, some processes and memory structures are needed to access the data and manage the database. The combination of Oracle (background) processes and memory buffers is called an Oracle **instance**.

Every running Oracle database is linked to an Oracle instance. Moreover, every Oracle database needs its own instance.



Hint: Using Real Application Clusters (RAC), one database is served by two or more instances.

SGA

Every time an Oracle instance is started, a shared memory region called the **System Global Area (SGA)** is allocated. The SGA allocated by an Oracle instance can only be accessed by the processes of this instance. This means that each instance has its own SGA. The SGA contains copies of data and control information for the corresponding Oracle instance. When the instance is stopped, the SGA is deallocated.

processes

Every time an Oracle instance is started, Oracle background **processes** are started. When an instance shuts down, the processes are stopped.



Hint: In a UNIX environment, Oracle processes are visible as operating system processes. On Windows platforms on the other hand, they are implemented as threads, which run within an Oracle operating system process *oracle.exe*. This means that they do not appear in the operating system process display.

system identifier (DBSID)

Every database is uniquely identified in the network by its **system identifier**. On SAP systems, the system identifier must consist of exactly three characters, the first of which must be an uppercase letter, while the other two can be uppercase letters or digits. Because the term system identifier is also used for SAP systems, we consequently distinguish between the database system identifier (referred to as DBSID) and the SAP system identifier (referred to as SAPSID).

Database architecture

Oracle Instance and Database: Architecture Overview

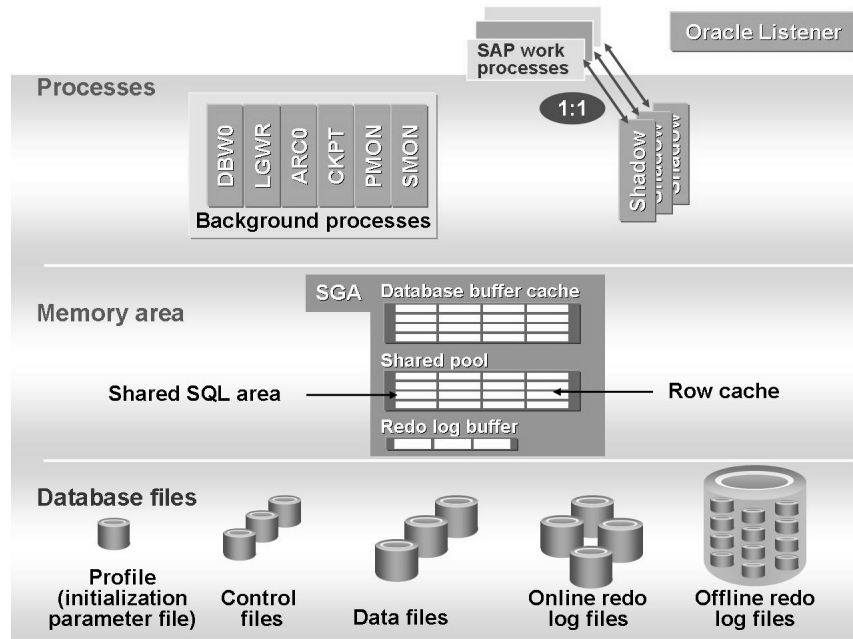


Figure 2: Architecture Overview

After an Oracle instance is started, a special process, the **Listener** allows the database clients and the instance to communicate with each other.



Note: The listener process is not part of an Oracle instance; it is part of networking processes that work with Oracle.

In SAP installations, dedicated servers are used. When a work process makes a request to connect to the database, the listener creates a dedicated server process and establishes an appropriate connection.

- The separate server process created on behalf of each work process (generally, for each user process) is called a **shadow process**.
- To handle database requests from several SAP system users, a work process communicates with its corresponding shadow process.
- When a work process has lost its connection to the database system, it automatically reconnects after the database server is available again and a database request is to be processed.

Oracle **background processes** perform various tasks required for the database to function properly.

Databases are stored in data files on disks. To accelerate read and write access to data, it is cached in the **database buffer** cache in the SGA. .

The Oracle database management system holds the executable SQL statements in the **shared SQL area** (also called the **shared cursor cache**), which is part of the **shared pool** allocated in SGA. Another part of the shared pool called the **row cache** caches Oracle data dictionary information.

Caching of Data

Databases are stored in data files on hard disks. However, data is never processed on the disks themselves. No matter whether a database client just needs to read some data or wants to modify it, it is first copied by the associated shadow process from the disk to the database buffer cache in the system global area (if it was not already there).

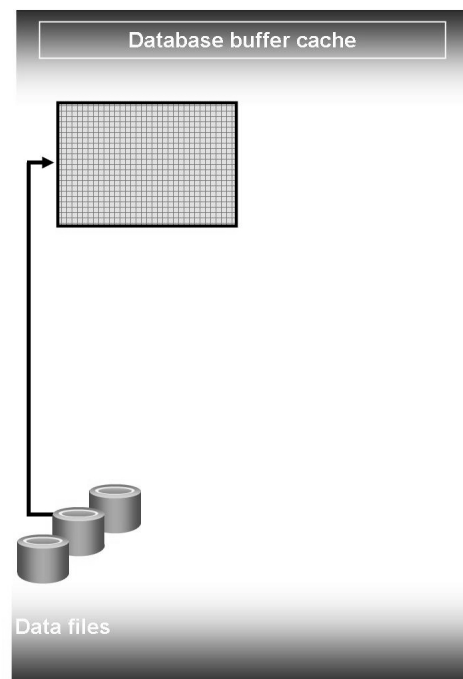


Figure 3: Oracle Architecture: Caching of Data

Data is always cached if the data is accessed for the first time after an Oracle instance is started. But as all users concurrently connected to the instance share access to the database buffer cache, copies of data read from data files into the buffer cache can be reused by any user.

The smallest logical unit that Oracle uses for copying data between data files and the buffer cache, as well as for managing data in the cache, is the **data block**.

- The size of an Oracle data block can generally be chosen during the creation of a database. In SAP installations, however, the block size is always 8 KB.
- For performance reasons, the physical allocation unit size on disks where you store Oracle files should also be 8 KB.

Oracle always tries to keep the most recently used data blocks in the buffer cache. Depending on the size of the buffer cache, it may sometimes be necessary to overwrite the least recently used data blocks in the buffer cache.

Writing of Modified Data

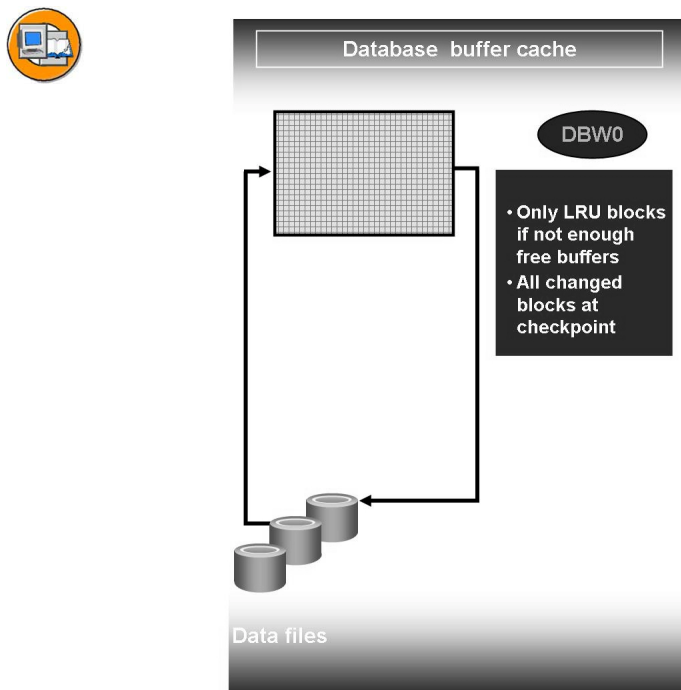


Figure 4: Oracle Architecture: Writing of Modified Data

Any changes to Oracle data (inserts, updates, and deletions), are always performed in the buffer cache. An Oracle shadow process itself never copies modified data blocks ('dirty blocks') from the buffer cache to the disk. This is the task of a special Oracle background process called the **database writer**.

The database writer process writes dirty blocks to disk in the following situations:

- Buffers in the buffer cache that contain dirty blocks (dirty buffers) may not be reused until these blocks have been copied back to disk. When a shadow process needs to copy data from disk to the buffer cache, it first scans the cache for non-modified, reusable buffers. If the number of scanned buffers reaches a certain threshold, the shadow process signals the database writer to start writing some of the modified blocks to disk. The database writer then copies those dirty blocks that are on the list of least recently used blocks (LSU list), thus making them free.
- At specific times, **all** modified buffers in the SGA are written to data files by the database writer. This process is called checkpoint, and **checkpoint process** (CKPT) triggers the database writer to perform the process.

Using the concept of deferred writes rather than immediate writes improves efficiency because in many cases, several changes on the same block are performed before the block is copied to disk. Also, the database writer performs multiblock writes in a batch(ed) style to increase I/O efficiency.



Hint: Although one database writer process (DBW0) is sufficient for most systems, additional database writer processes (DBW1 and so on) may be configured in exceptional cases. See SAP Notes 124361 and 191463 for more information.

Logging of Modifications

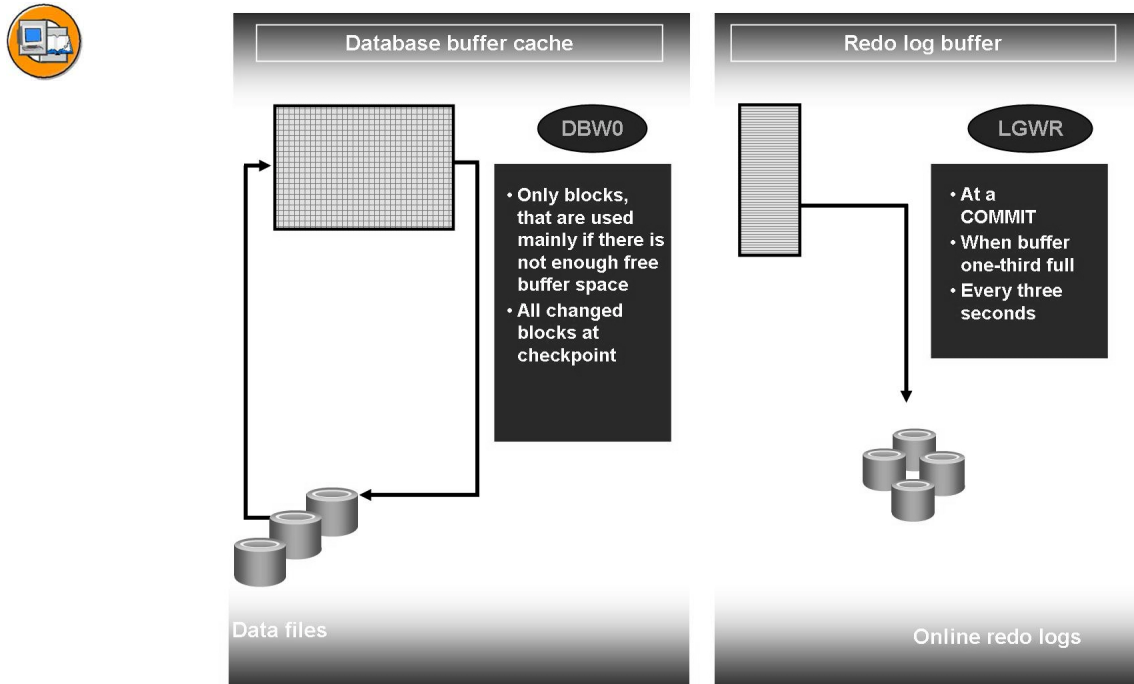


Figure 5: Oracle Architecture: Logging of Modifications

Because of deferred writes, a mechanism is needed to prevent data loss and also to avoid data inconsistencies in case of a crash of any system component (disk, Oracle instance, or server).

Generally, each RDBMS logs all data changes in a log area, which is written to disk at appropriate times. The log is generally written to disk after the commit of a database transaction so that all data block changes are logged.

A database transaction is a logical unit of work for the database server (LUW - logical unit of work), which is always treated as an “atomic” unit, meaning it must either be processed completely or not at all.

To achieve data consistency and read consistency, Oracle maintains **redo entries** for roll forward or redo recovery for example after a crash, and **undo entries** to roll back uncommitted transactions.

redo entries

Redo entries contain the information necessary to reconstruct, redo, or roll forward changes made to the database by SQL statements within a committed transaction. Redo entries contain the new values of the modified data, also called **after images**.

Parallel to changes made in data blocks, Oracle shadow processes write redo entries into the **redo log buffer**. This is a circular buffer in the system global area that temporarily records all changes (both uncommitted and committed) made to the database. The Oracle background process **log writer** (LGWR) then writes contiguous portions of the redo log buffer sequentially to an **online redo log file** (or group of files) on disk.


The online redo log consists of four (or more) online redo log files. Redo entries in the online redo log are used for the database recovery if necessary (in order to redo changes).

The log writer writes entries from the redo log buffer to the online redo log:

- When any transaction commits (the LGWR also writes a special commit record for the corresponding transaction in this case)
- Every three seconds
- When the redo log buffer is one-third full
- When the database writer is about to write modified buffers from the block buffer to disk and some of the corresponding redo records have not yet been written to the online redo log files (write-ahead logging).

This algorithm ensures that space is always available in the redo log buffer for new redo records.

When a user (a work process) commits a transaction, the transaction is assigned a **system change number** (SCN) by the database system. Oracle records the SCN along with the transaction's redo entries in the redo log.

 **Note:** Because Oracle uses write-ahead logging, DBW0 does not need to write data blocks when a transaction commits.

undo entries

Undo entries contain the information necessary to undo, or roll back, any changes to data blocks that have been performed by SQL statements, which have not yet been committed. Undo entries contain the old values of the modified data, also called **before images**.

Oracle stores undo information (old values of modified data), called before images in a special undo segment, separate from the redo log.

The Oracle undo space consists either of an undo tablespace (this solution is called automatic undo management — only the undo tablespaces must be created) or of **rollback segments** (**manual** undo management — rollback segments must be allocated in a tablespace and managed).

The undo information of a transaction is retained in the undo space at least until the end of the transaction. It may be overwritten only after the transaction has been committed.

During database recovery, Oracle first applies all changes recorded in the redo log (which includes the recovery of changes in the undo space), and then uses the undo information to roll back any uncommitted transactions.

Moreover, Oracle can use the undo entries for other purposes, including reading of snapshots of consistent data (accessing before images of blocks changed in uncommitted transactions).

Log switch

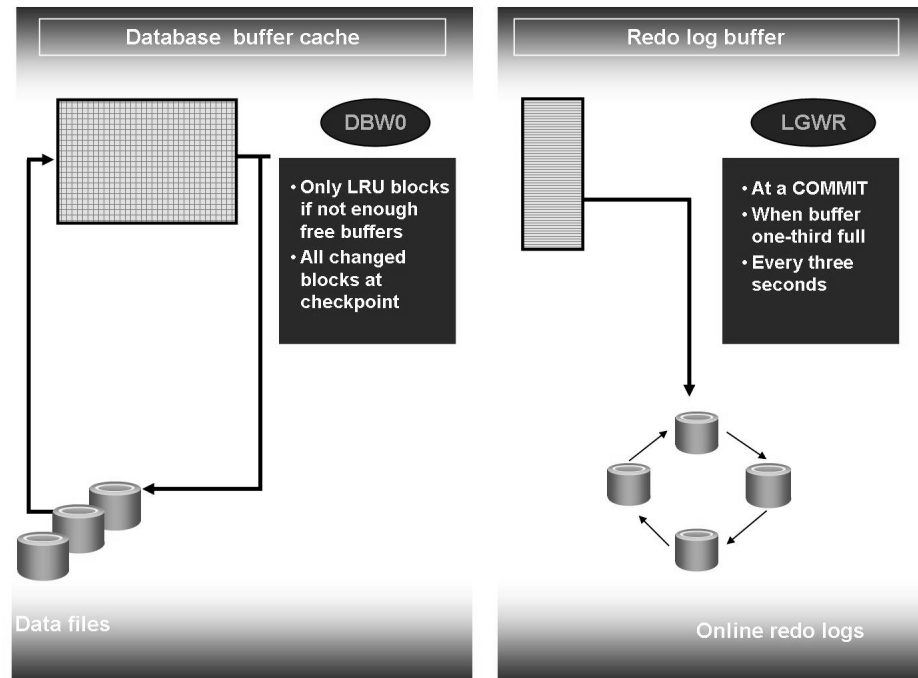


Figure 6: Oracle Architecture: Log switch

Oracle redo log files do not dynamically grow when more space is needed for redo entries; they have a fixed size (on SAP systems, typically 50 MB). When the current online redo log file becomes full, the log writer process closes this file and starts writing into the next one. This is called a **log switch**.

- The predefined collection of online redo log files (four in our example) is used in a cycle.
- At every log switch, Oracle increases the **log sequence number (LSN)**. Through the LSN, Oracle automatically creates a sequential numbering of redo logs.
- The online redo log file into which the LGWR is currently writing is called the **current** online redo log file.

Control files

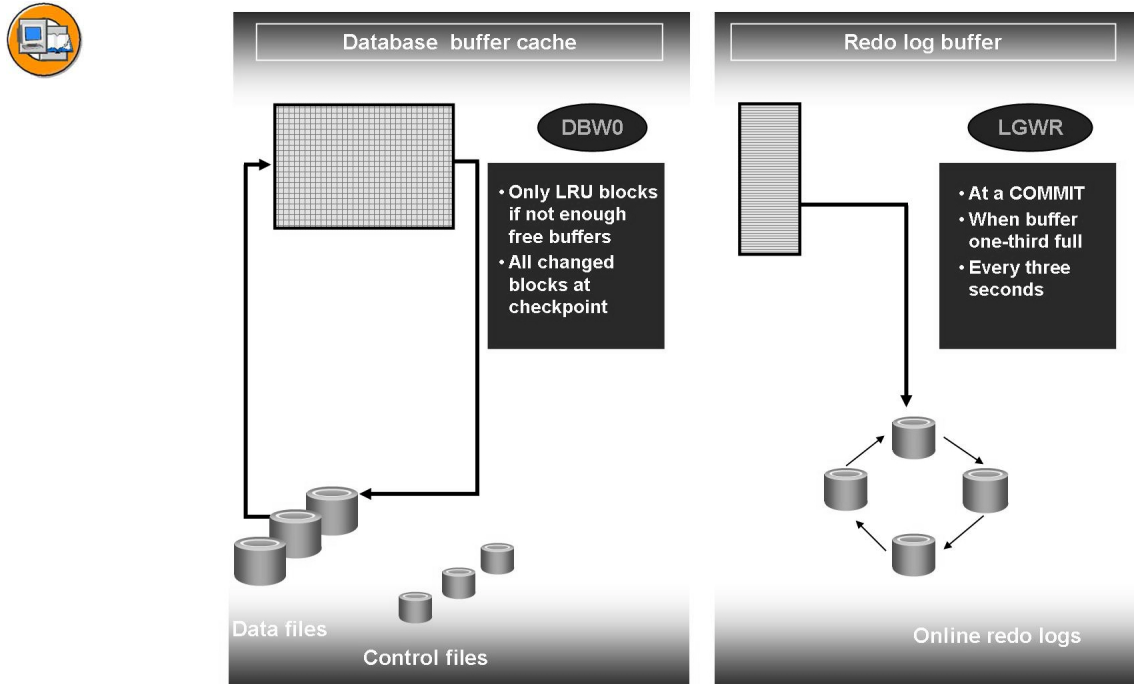


Figure 7: Oracle Architecture: Control files

Every Oracle database has a **control file**, which is a small binary file necessary for the database to start and operate successfully. A control file contains entries that specify the physical structure and state of the database, such as tablespace information, names and locations of data files and redo log files, or the current log sequence number. If the physical structure of the database changes (for example, when creating new data files or redo log files), the control file is automatically updated by Oracle.

- Control files can be changed only by Oracle. No database administrator or any other user can edit the control file directly.
- After opening the database, the control file must be available for writing. If for some reason the control file is not accessible, the database cannot function properly.
- Oracle control files can be mirrored for security reasons. Several copies can be stored at different locations; Oracle updates these at the same time. In SAP installations, the control file is stored in three copies, which must be created on three physically separate disks.

Checkpoints

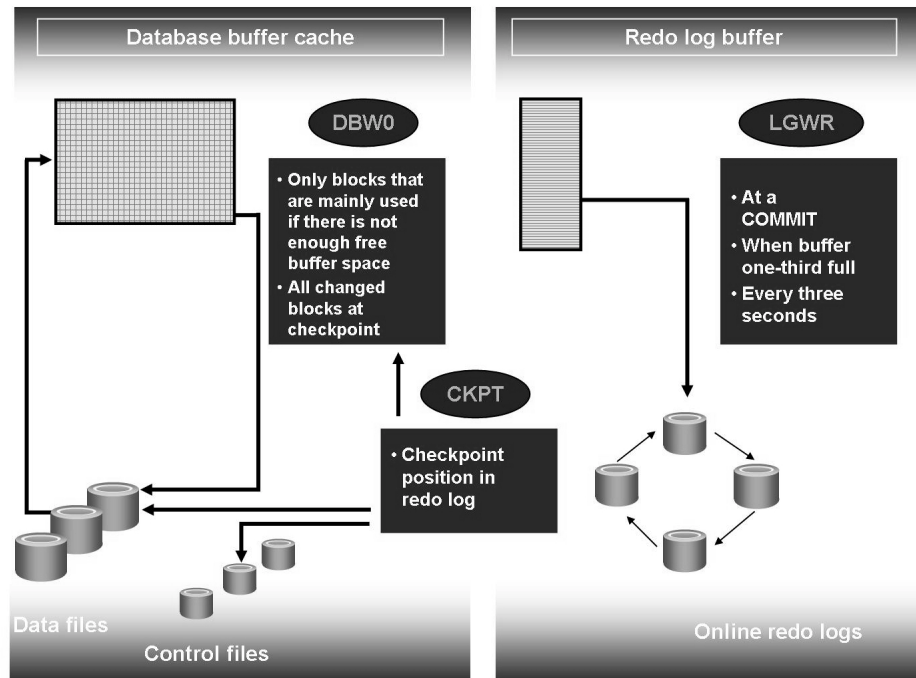


Figure 8: Oracle Architecture: Checkpoints

The term **checkpoint** has at least two different meanings.

The checkpoint is the point at which the database writer writes all changed buffers in the buffer cache to the data files.

The database writer (DBW0) receives a signal at specific times from the **background checkpoint process** (CKTP) to perform this action. DBW0 then copies all buffers that were dirty at that moment to disk. Before DBW0 finishes this job, other blocks in the buffer cache can become dirty.

After the checkpoint event has finished, the oldest dirty buffer in the buffer cache determines a point in the redo log from which instance recovery must begin if a crash occurs. This log position is also called a **checkpoint**.

The task of the checkpoint process is not only to activate the database writer, and it does not write blocks to disk (because this is the task of DBW0):

- It also writes checkpoint information to the data file header.
- It writes information about the checkpoint position in the online redo log into the control file.

The information about the checkpoint position in the online redo log in the control file is needed for instance recovery. It tells Oracle that all redo entries recorded before this point are not necessary for database recovery, as they were already written to data files.

The frequency of checkpoints is one of the factors that influences the time required for the instance to recover from a failure. The less frequent the checkpoints, the longer the time the instance needs for recovery.

- A checkpoint always occurs at a log switch.
- Frequency of checkpoints can be specified with help of Oracle parameters. In SAP installations, these parameters have values such that they are effectively not used, and checkpoints occur only at log switches.

Database Recovery

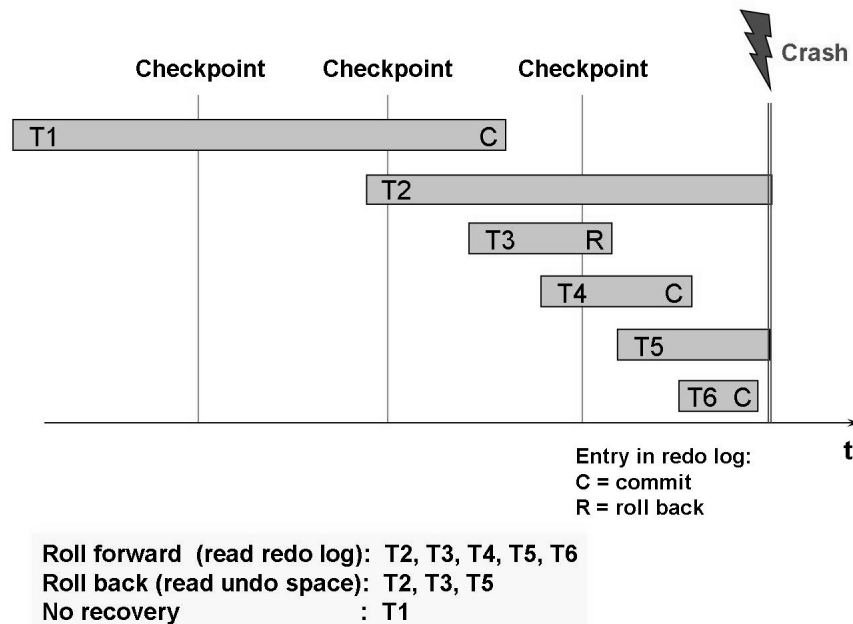


Figure 9: Oracle Architecture: Database Recovery

Online redo logs play an important role when starting an Oracle instance and opening the database, especially after a crash or when the instance was not shut down “cleanly”. In this situation, Oracle recognizes that the database was not properly shut down and automatically initiates database recovery (also called instance recovery).

The automatic recovery at restart consists of two phases:

- Starting at the checkpoint position, redo entries are read from the online redo log and transactions are “reprocessed” (roll forward). This includes the roll forward of changes in the undo space.
- For every transaction that was either uncommitted at the time of the crash or rolled back explicitly before the crash (so that there is no commit entry for it in the redo log), a rollback is performed with the help of before images read from the undo space. Oracle ensures that this is always possible because it never deletes undo entries of open transactions from the undo space.

The result is a consistent database containing only changes committed before the crash.

In the example in the above figure:

- Transaction 1 is not relevant for redo/undo, as it was committed at the time of the last checkpoint. Changes to this transaction were written to disk at the last checkpoint.
- Transactions T2, T3, T4, T5, and T6 are redone, as they caused changes in the database after the last checkpoint. However, among these, only the changes to T4 and T6 are committed, which means only these changes are persistent.
- Transactions T2, T3, and T5 are rolled back.

Redo Log Mirroring

From a data security point of view, the online redo logs are one of the most critical areas in an Oracle server. If you lose them in a crash, a complete recovery of the database is not possible, and the result is a loss of some data.



Caution: Online redo logs must always be mirrored, meaning that two or more copies of each redo log must be maintained on different disks.

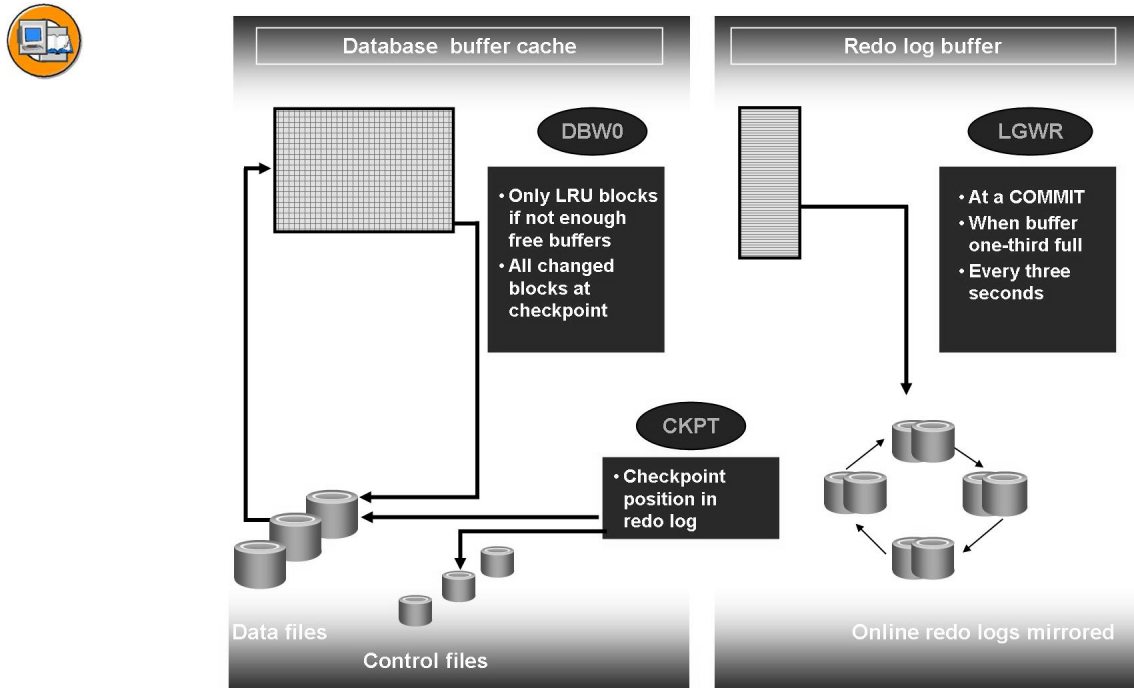


Figure 10: Oracle Architecture: Redo Log Mirroring

Oracle itself can mirror online redo logs. This feature is used in SAP installations by default, so that there is no need for a software or hardware RAID solution (redundant array of independent disks) solution. On the other hand, from the data security point of view, it does not matter which solution you choose. Even a combination of both Oracle and RAID mirroring is feasible to minimize the risk of losing an online redo log.

Archiving

As the online redo log is limited in size and cannot grow automatically, Oracle must overwrite old redo entries to write new ones.

Only the oldest redo entries up to the checkpoint position in the log, which corresponds to data changes that have already been written to data files, can be overwritten. This ensures that automatic instance recovery after a crash is always possible.

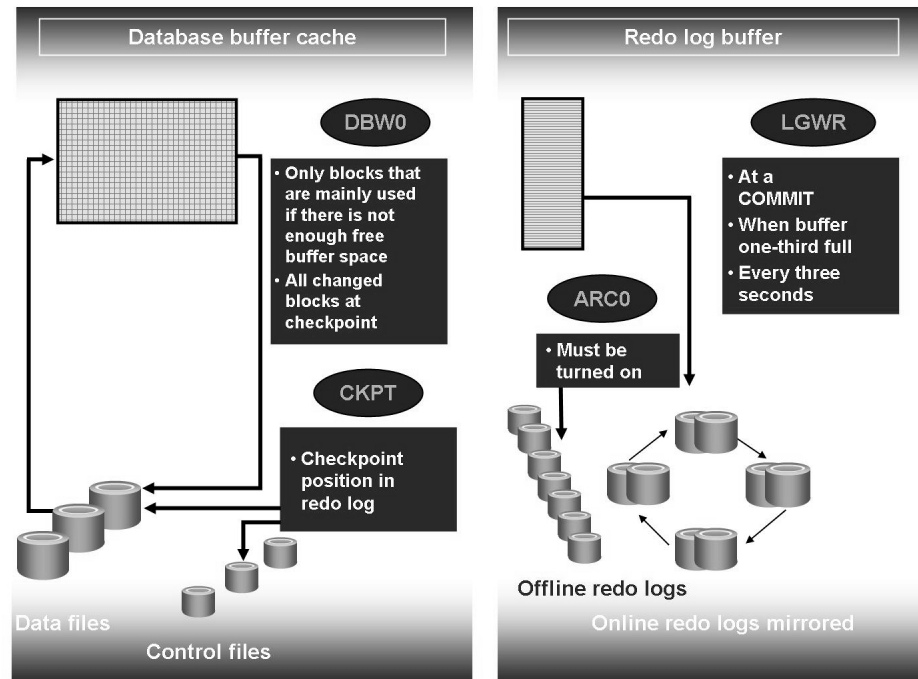


Figure 11: Oracle Architecture: Archiving

However, in a situation where you must restore data files after a disk crash and recover them manually (usually to the state the data had at the point of the crash), you need both a database backup and all the redo information written after this backup. In an SAP system, log switches occur every few minutes so that online redo log files are reused very frequently. To prevent loss of redo information, it must be copied from online redo log files to a safe location before overwriting. This is the task of a special Oracle background process called **archiver** (ARC0).

Archiving must be explicitly activated, through turning on the ARCHIVELOG mode of the database and setting the Oracle instance parameter LOG_ARCHIVE_START to TRUE.

When LOG_ARCHIVE_START is TRUE, the archiver process automatically starts when an Oracle instance is started. When the ARCHIVELOG mode of the database is turned on, the archiver process automatically copies a newly written online redo log file – after a corresponding switch to the next online redo log file – to an offline redo log file, and overwriting old redo log entries in online redo logs is not allowed before the entries have been copied to offline redo logs. Once an offline redo log file has been successfully created as a copy, the corresponding online redo log file is released to be overwritten with new log information. The directory where offline redo log files are created can be specified through an Oracle parameter.



Caution: Archiving must be activated in productive systems. Moreover, offline redo log files should be stored on a mirrored disk to prevent loss of redo information. A RAID system can be used for this purpose.

In an SAP system, the activation of archiving is the default setting. A deactivation of archiving by changing the database log mode to NOARCHIVELOG (required, for example, during a system upgrade) is supported by the SAP tool BRSPACE.



Caution: If you lose a disk containing offline redo logs and data files after a crash, complete recovery is no longer possible. Therefore, offline redo logs and data files should be stored on different disks.

Other Background Processes

There are two more background processes that always run in an Oracle instance:

System Monitor (SMON)

- Performs recovery at instance startup, if necessary
- Writes alert log information if any other instance process fails
- Cleans up temporary segments that are no longer in use

Process Monitor (PMON)

- Monitors shadow processes
- In case a client process crashes, PMON rolls back its non-committed transaction, stops the corresponding shadow process, and frees resources that the process was using.

Oracle Directory Structure in SAP

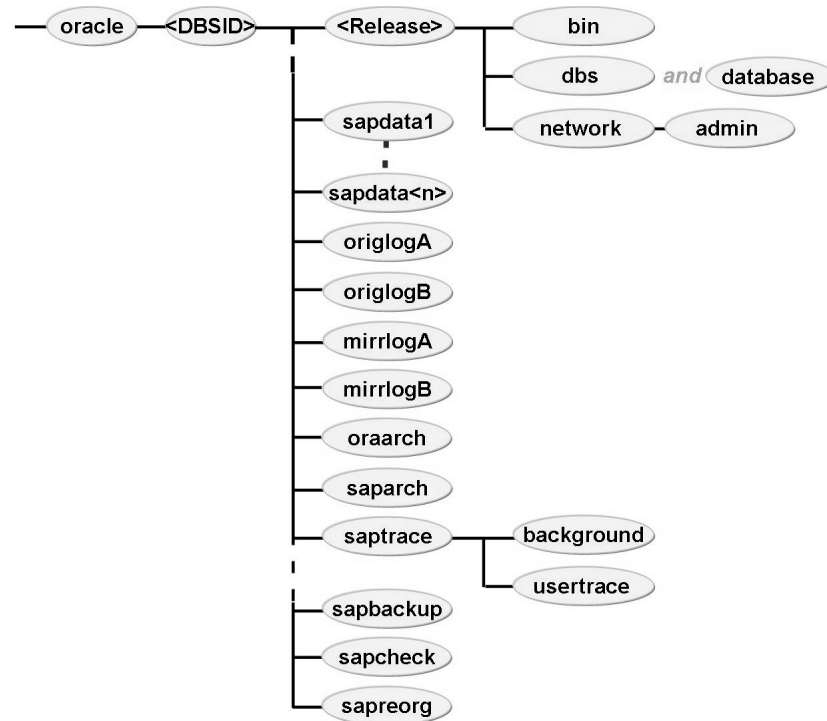


Figure 12: Oracle Directory Structure on SAP Systems

The Oracle directory (folder) and file names are standardized in SAP environments. Directories (folders) are always created with the same structure and naming convention during the installation. This structure may not be changed, and the naming conventions must be observed, as SAP tools for Oracle administration rely on it.

Various parts of the Oracle directories and files must be physically separated from each other for performance and data security reasons. On UNIX systems, the Oracle directories appear as a tree structure because the file systems created on the physical disks are mounted on directories. On Windows, however, you will have several \oracle\<DBSID> folders on different disks with different drive letters.



Hint: On UNIX, the <Release> subdirectory under /oracle/<DBSID> also contains information whether you use a 32-bit or 64-bit Oracle version, for example, /oracle/<DBSID>/102_64 for a 64-bit Oracle 10.2.



Directory	Contents	File name examples
<Release>		
bin	Oracle executables	
db ^s or database	SAP and Oracle profiles	init<DBSID>.ora, spfile<DBSID>.ora, init<DBSID>.sap
Network		
admin	Listener or client configuration files	listener.ora, tnsnames.ora
sapdata1	Data files	<DBSID>.DATA1, SYSTEM.DATA1
.		ROLL.DATA1, ctrl<DBSID>.dbf
sapdata<n>		
origlogA	Online redo log files	log_g11m1.dbf, log_g13m1.dbf
origlogB	Online redo log files	log_g12m1.dbf, log_g14m1.dbf
mirrlogA	Online redo log files	log_g11m2.dbf, log_g13m2.dbf
mirrlogB	Online redo log files	log_g12m2.dbf, log_g14m2.dbf
oraarch	Offline redo log files	<DBSID>arch1_<LSN>.dbf
saparch	BRARCHIVE logs	arch<DBSID>.log
saptrace		
background	Oracle alert file	alert<DBSID>.log
usertrace	Trace files of server processes	<DBSID>_ora_<PID>.trc
sapbackup	BRBACKUP / BRRESTORE / BRECOVER	
logs		
sapcheck	BRCONNECT logs	
sapreorg	BRSPACE logs, default compression directory	

Figure 13: Oracle Directories and Files on SAP Systems

db^s (on UNIX) or database (on Windows)

- The Oracle profile `init<DBSID>.ora` or `spfile<DBSID>.ora` holds the Oracle instance configuration parameters
- The profile `init<DBSID>.sap` holds configuration parameters for administration tools BR*Tools

sapdata<n>

Contains the data files of the tablespaces

origlogA/B, mirrlogA/B

Online redo log files reside in the `origlog` and `mirrlog` directories: Log file numbers 1 and 3 and their mirrors in `origlogA` and `mirrlogA`, log file numbers 2 and 4 and their mirrors in `origlogB` and `mirrlogB`, respectively

oraarch

Offline redo log files are written to the `oraarch` directory; their names are specified with help of Oracle instance configuration parameters, so the name `<DBSID>arch1_<LSN>.dbf` is just an example.

saptrace

Oracle dump files are written in the directory `saptrace`. The Oracle alert log `alert_<DBSID>.log` occurs in the directory `saptrace/background`. Traces of Oracle shadow processes are written to the directory `saptrace/usertrace`.

saparch

Stores the logs written by the SAP tool BRARCHIVE

sapbackup

Stores logs written by the SAP tools BRBACKUP, BRRESTORE, and BRRECOVER

sapreorg

BRSPACE creates logs for its different functions here

sapcheck

BRSPACE creates logs for its different functions here

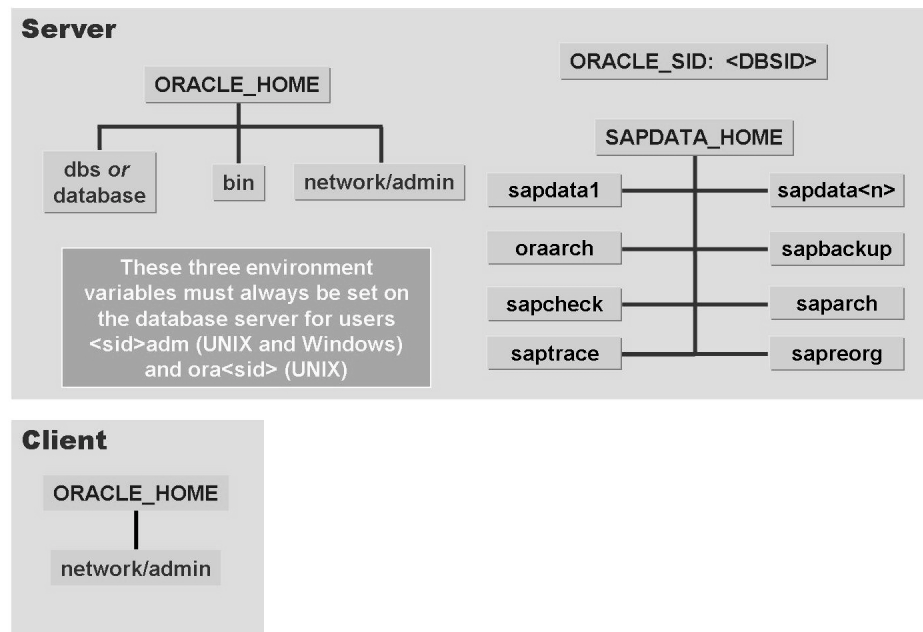
Oracle Directories and Environment Variables

Figure 14: Oracle Directories and Environment Variables

On the database server, the environment variables `ORACLE_SID`, `ORACLE_HOME`, and `SAPDATA_HOME` must always be set for the user `<sapsid>adm`, as well as for the user `ora<dbsid>` on a UNIX platform.

ORACLE_SID

This is the system ID of the database instance (DB SID).

ORACLE_HOME

This is the home directory of the Oracle software. More precisely, `ORACLE_HOME` points to the directory that contains subdirectories `bin`, `db`s (or `database`), and `network`. This means in particular that the Oracle profile `init<DBSID>.ora` or `spfile<DBSID>.ora` is always located in `$ORACLE_HOME/db`s (in `%ORACLE_HOME%\database` on Windows).

SAPDATA_HOME

Points to the directory in which the database files are stored.



Hint : The location of the control files and of the offline redo logs is configured in the Oracle profile `init<DBSID>.ora`; the location of all other files (data files, online redo logs, and so on) is stored in the database itself. Therefore, `SAPDATA_HOME` is mainly used by BR*Tools to offer suitable directories, for example, when new tablespaces or data files need to be created.

other variables

There are also other variables you can set if the corresponding directories do not have any subdirectories of `SAPDATA_HOME`. This is often the case on Windows systems owing to the different drive letters: `SAPARCH`, `SAPBACKUP`, `SAPCHECK`, `SAPREORG`.

On an Oracle client (especially on every SAP application server), the variable `ORACLE_HOME` must also be set so that connection information can be found in `$ORACLE_HOME/network/admin`.

In a Unix environment, the environment variables `ORA_NLS10` are also set for the user `ora<dbsid>`. The default value for `ORA_NLS10` is `$ORACLE_HOME/nls/data`. Since `ORACLE_HOME` is set, `ORA_NLS10` does not need to be set.

For the user `<sapsid>adm`, the environment variable `ORA_NLS10` is not set or must not be set. The Oracle instant client downloads the NLS data from a dynamic library (NLS library), which is stored in the Instant Client directory.

See SAP Note 830578 for information on how to set this variable correctly for Oracle 10g. For earlier Oracle releases, see SAP Note 180430 and the other SAP Notes referenced there.

Oracle Real Application Clusters (RAC)

To improve performance, increase throughput, and deliver high availability at the same time, install your SAP system in a real application clusters (RAC) environment. RAC overcomes the restrictions of normal failover solutions with:

- Concurrent processing
- Load balancing
- Fast and reliable detection of a node or network failure
- Fast recovery

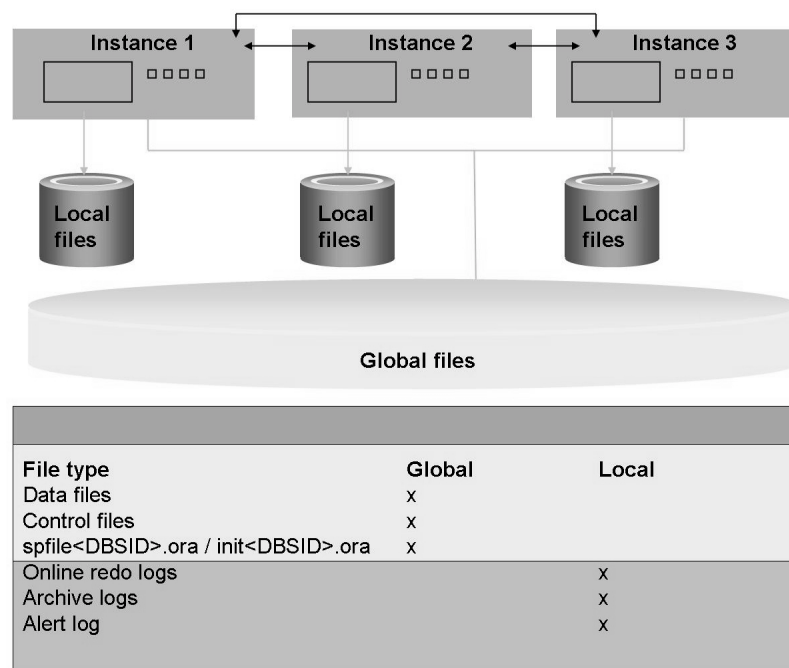


Figure 15: Real Application Clusters

While with a standard cluster solution you use just one active database instance, RAC makes it possible to use several instances simultaneously. All active instances execute transactions against a shared database. To provide data consistency and data integrity, real application clusters coordinates each instance's access to the shared data.

- In real application clusters, the Transport Network Service (TNS) listener files provide automated load balancing across all nodes in the cluster.
- Moreover, the RAC's load balancing feature automatically adjusts for cluster configuration changes. For example, if you add a node to your cluster database, Oracle updates all the listener files in the cluster with the new node's listener information.

There is no need to make code changes to deploy applications on RACs if the applications ran on single-instance Oracle configurations. In particular, no adjustments in the SAP application are needed for RACs.

Real application clusters require that all nodes have simultaneous access to the shared disks to give the instances concurrent access to the database. The implementation of the shared disk subsystem is based on the operating system chosen; you can use either a cluster file system or place the files on raw devices. However, cluster file systems greatly simplify the installation and administration of RACs.

Apart from using a shared database, real application clusters coordinates the buffer caches of multiple instances on different nodes. This optimizes performance and expands the effective memory to be nearly equal to the sum of all memory in your cluster database.

RACs also support all Oracle backup and archiving features that are available in single-instance Oracle databases. This includes both online and offline backups of either an entire database or individual tablespaces.

Migration to Oracle real application clusters is relatively easy, as no unloading and loading of data is necessary.

Exercise 1: Oracle Environment Variables

Exercise Objectives

After completing this exercise, you will be able to:

- Explain the importance of Oracle environment variables

Business Example

You log on to the database server and call Oracle tools. Strange error messages appear, because environment variables are not set or set incorrectly. You want to know the reason for these error messages.

Task:

Change Oracle environment variables and see the consequences

1. Log on to the database server with the logon information provided by your instructor.
2. Check the environment variables set, especially Oracle variables starting with ORA.
3. Change variable ORACLE_HOME to any other path, then call the Oracle tool TNSPING to ping your <DBSID>. Check the output and explain the reason for any strange output.

Solution 1: Oracle Environment Variables

Task:

Change Oracle environment variables and see the consequences

1. Log on to the database server with the logon information provided by your instructor.
 - a) Use the information provided by your instructor.
2. Check the environment variables set, especially Oracle variables starting with ORA.
 - a) Use the `set` command to display environment variables. Note that paths, drive letters, and system IDs might be different than in the following output.

```
G:\oracle\T99>set
...
ORACLE_HOME=G:\oracle\DEV\102
ORACLE_SID=T99
...
SAPARCH=G:\oracle\DEV\saparch
SAPBACKUP=G:\oracle\DEV\sapbackup
SAPCHECK=G:\oracle\DEV\sapcheck
SAPDATA_HOME=G:\oracle\T99
SAPEXE=G:\usr\sap\DEV\SYS\exe\run
SAPLOCALHOST=twdf9999
SAPREORG=G:\oracle\DEV\sapreorg
SAPTRACE=G:\oracle\DEV\saptrace
```

3. Change variable ORACLE_HOME to any other path, then call the Oracle tool TNSPING to ping your <DBSID>. Check the output and explain the reason for any strange output.
 - a) Then call the command TNSPING without making any changes, and check whether your database can be accessed.

```
G:\oracle\T99>tnsping T99
```

```
TNS Ping Utility for 64-bit Windows: Version 10.2.0.2.0 -
Production on 26-NOV-2007 10:04:46
```

```
Copyright (c) 1997, 2005, Oracle. All rights reserved.
```

Continued on next page

Used parameter files:

g:\oracle\DEV\102\network\admin\sqlnet.ora

Used TNSNAMES adapter to resolve the alias

Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS =
(PROTOCOL = TCP) (HOST = TWDF9999) (PORT = 1527))) (CONNECT_DATA =
(SERVICE_NAME = T99))) OK (20 msec)

- b) Change the variable ORACLE_HOME, and call the command TNSPING again.

G:\oracle\T99>set ORACLE_HOME=G:\

G:\oracle\T99>tnsping T99

TNS Ping Utility for 64-bit Windows: Version 10.2.0.2.0 -
Production on 26-NOV-2007 10:13:09

Copyright (c) 1997, 2005, Oracle. All rights reserved.

Message 3511 not found; No message file for product=NETWORK,
facility=TNSTNS-03505: Message 3505 not found; No message file
for product=NETWORK, facility=TNS

- c) Oracle internally works with message numbers, whose message texts are stored in a subdirectory of %ORACLE_HOME%. If ORACLE_HOME is not set correctly, texts cannot be displayed.

While TNSPING does not display a useful reason for the strange output, SQLPLUS does:

G:\oracle\T99>sqlplus /nolog
Error 6 initializing SQL*Plus
Message file sp1<lang>.msb not found
SP2-0750: You may need to set ORACLE_HOME to your Oracle software
directory

- d) Change the ORACLE_HOME variable back to your original value.

G:\oracle\T99>set ORACLE_HOME=g:\oracle\DEV\102



Lesson Summary

You should now be able to:

- Describe the architecture and the main components of an Oracle database
- Explain the basic concepts of the Oracle database
- Identify the file structure of an Oracle database in a SAP system

Lesson: Connecting to the database

Lesson Overview

In this lesson you will learn how SAP connects to an Oracle database over the network.



Lesson Objectives

After completing this lesson, you will be able to:

- Name the default operating system and database users
- Explain Oracle communication over a network (NETServices)
- Describe the function of the Oracle listener
- Start and stop the Oracle listener

Business Example

A database administrator changed the password of the default database user *SAP<SCHEMA-ID>* using Oracle commands. Afterward, the SAP system could not be started anymore. In this lesson, you will learn how to correctly change passwords.

Operating System and Database Users

To safeguard your SAP system, you must control user access on three different levels:

- Operating system
- database
- SAP system

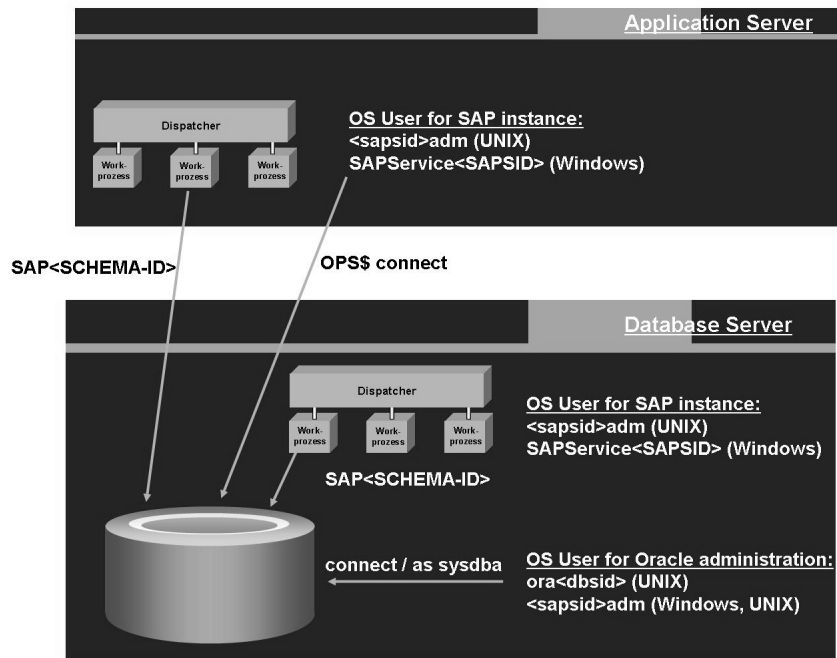


Figure 16: Operating System and Database Users

Database users are required in the SAP environment for two scenarios: First the SAP system itself connects to the database during operation, and second the database administrators connect to the database to perform administrative actions. The corresponding users and their assigned roles and privileges are explained in the following sections.

Oracle System Privileges



Operations in Oracle are controlled by system privileges and object privileges

Oracle system privileges: Examples

Privilege name	Operations authorized
CREATE SESSION	Connect to the database
CREATE TABLESPACE	Create a new tablespace for the database
ALTER SYSTEM	Issue "ALTER SYSTEM" statements
.....	

One or two special system privileges is required for database administration

Special system privileges

Privilege name	Operations authorized
SYSDBA	STARTUP, SHUTDOWN, CREATE DATABASE, ARCHIVELOG, BACKUP, RECOVERY, ...
.SYSOPER	As for SYSDBA but without CREATE DATABASE and without the ability to look at user data

The SYSDBA and SYSOPER system privileges allow access to a database instance even when the database is not open. Control of these privileges is totally outside of the database itself.

Figure 17: Oracle System Privileges

System privileges restrict actions performed by database users on the instance or database level. There are over 100 system privileges in Oracle.

Object privileges protect access at object level, so that you can set the right to send queries to tables or views (SELECT) for example, and perform DML operations (INSERT, UPDATE, DELETE).

The special system privileges SYSDBA and SYSOPER can be thought of as types of connections. In an SAP system, we use operating system authentication to connect to Oracle with the privileges SYSDBA or SYSOPER.

SYSDBA and SYSOPER give you special administrative privileges, enabling you to perform certain database operations for which authorizations cannot be granted in any other fashion.

Operating System Users and Groups

In SAP systems with Oracle, special operating system users created during the installation have privileges for administration and maintenance of the Oracle database on two levels:

- They can access Oracle instance directories and files and call database administration tools on operating system level.
- They can connect to the Oracle instance with special database users and either perform administrative work or maintain SAP objects and data in the database.



Operating System Users and Groups in an SAP System with Oracle

UNIX environment

OS user	Oracle-relevant OS group	Privileges in oracle
ora<dbsid>	dba oper	Full administration of all instances Restricted administration of all instances
<sapsid>adm	dba oper	Full administration of all instances Restricted administration of all instances

Windows 2000/2003 environment

OS user	Oracle-relevant OS group	Privileges in oracle
<sapsid>adm	ORA_<DBSID>_DBA ORA_<DBSID>_OPER ORA_DBA	Full administration of instance Restricted administration of instance Full administration of all instances
SAPService<SAPSID>	ORA_<DBSID>_DBA ORA_<DBSID>_OPER ORA_DBA	Full administration of instance Restricted admin of the instance Full administration of all instances

Figure 18: Operating System Users and Groups in an SAP System with Oracle

Oracle is able to move the database security mechanism to the operating system level using certain mappings between OS users and DB users, or between OS groups and system privileges. For example:

- Members of operating system group dba (UNIX) or ORA_DBA or ORA_<DBSID>_DBA (Windows) can connect to the Oracle instance with system privilege SYSDBA and perform administrative work there.
- Members of operating system group oper (UNIX) or ORA_<DBSID>_OPER (Windows) can connect to the Oracle instance with system privilege SYSOPER and perform corresponding administrative work there.



Hint: While a member of the Windows local group ORA_<DBSID>_DBA can connect to and administer the instance <DBSID>, a member of ORA_DBA is able to do this in any Oracle instance installed on the corresponding host.

Oracle Database Users

Every Oracle database contains two administrative user accounts, SYS and SYSTEM, which are automatically created during installation and assigned the database role DBA.



Standard Database Users in Oracle

User name	Default password	Description
SYS	* CHANGE_ON_INSTALL	Owner of the database's data dictionary tables and views; can perform database administration; has privileges to access and modify all database tables and data.
SYSTEM	* MANAGER	Can perform database administration; has privileges to access all database tables and data, but cannot modify data dictionary tables

* Starting with SAP Web AS 6.40 the password for the database users are defined during installation.

Figure 19: Standard Database Users in Oracle

SYS is the user with the most privileges in an Oracle database:

- All tables and views of the database's data dictionary are stored in the schema SYS. These tables and views are important for operating Oracle. They should therefore never be modified by a user or database administrator. You should not create tables in the schema of the SYS user.
- SYS is granted some additional privileges over those of the DBA role, and can access and modify all data in the database.



Note: A schema is a collection of database objects belonging to a user as owner. A schema always has the same name as the owner.

SYSTEM is a username defined by Oracle for the creation of additional internal tables and views that display administrative information. Although SYSTEM can access all database tables, it has no privilege to change Oracle data dictionary tables.

In an SAP installation:

- SYSTEM is additionally assigned the database role SAPDBA to allow BR*Tools access to certain tables of the SYS schema.
- SYSTEM is the default user when SAP tools are called for Oracle administration for creating a connection to the database.



Hint: Oracle user and role names, as well as user passwords, are not case sensitive unless you use them as strings enclosed in quotation marks.



Database Users in Oracle Created by SAP		
OS platform independent		
User name	Default password	Description
SAP<SCHEMA-ID> or SAPR3	SAP	All SAP objects in Oracle are created here schema of this user; no privileges for administration of the database Assigned role: SAPCONN
UNIX environment		
User name		Description
OPSS<SAPSID>ADM		Assigned role: SAPDBA (but not DBA)
OPSSORA<DBSID>		Assigned role: SAPDBA (but not DBA)
Windows 2000/2003 environment		
User name		Description
OPSS<DOMAIN>\<SAPSID>ADM		Assigned role: SAPDBA (but not DBA)
OPSS<DOMAIN>\SAPSERVICE<SAPSID>		Assigned role: SAPDBA (but not DBA)
Operating System Authentication		
Oracle parameters		
REMOTE_OS_AUTHENT=TRUE OS_AUTHENT_PREFIX=OPSS\$		
Operating system user		Oracle database user
<USERNAME>		OPSS<USERNAME>
OPSS\$ users have no passwords, they are <i>identified externally</i>		

Figure 20: Database Users in Oracle Created by SAP

The SAP installation always creates the Oracle user SAP<SCHEMA-ID> (or SAPR3 up to SAP Basis4.6D), where SCHEMA-ID is, in most cases, identical with SAP SID. All tables and indexes of the corresponding SAP system belong to the schema of this database user. However, SAP<SCHEMA-ID> does not have privileges to perform administrative tasks on the database; it is not assigned the database roles DBA or SAPDBA.



Caution: During the installation of an Oracle database for an SAP system, you are asked to specify passwords for the users SYS, SYSTEM, and SAP<SCHEMA-ID>. If you do not enter anything, the users are assigned default passwords. In this case, you must change the passwords immediately after the installation is completed. If you do not change the passwords, your system is not sufficiently secure.

Other users created in the Oracle database by SAP make use of an Oracle feature called **operating system authentication**. If the user OPS\$<USERNAME> is defined as identified externally at the database level, it has no password and the operating system user <USERNAME> can connect to the database without authentication, assuming that the following two Oracle parameters are set:

- REMOTE_OS_AUTHENT=TRUE (allows remote operating system authentication for OS users with an OPS\$ user on any computer in the network from which the database is accessible)
- OS_AUTHENT_PREFIX=OPS\$

These are the default values of the parameters in an SAP system, so normally you do not have to change them.

On a Windows platform, <USERNAME> used in the definition of the OPS\$ user includes the name of the Windows domain from which the operating system user originates (or the host name if it is a local user).

Oracle Database Roles

Within the database, system and object privileges can be pooled to database roles. If you assign a database role to a database user, the user will be granted all privileges included in the role.

There are only a few predefined database roles in Oracle, the most important of which is DBA. The DBA role contains all privileges necessary for database administration (the privileges carrying the flag ADMIN OPTION). However, the DBA role does not enable you to fully administer the Oracle instance; for example, to be able to start and shut down the instance, you also need the system privilege SYSDBA or SYSOPER.



Privileges are Grouped and Granted to Users Through Database Roles

Predefined Oracle database roles

Role	Description
DBA	Contains all system and object privileges needed for administration of the database, however does <u>not</u> include the SYDBA or SYSOPER system privileges.
SELECT_CATALOG_ROLE	Provides SELECT privilege on objects in the data dictionary.
.....	

Database roles created by SAP

Role	Description
SAPDBA	Contains system privileges such as ALTER SYSTEM, ALTER DATABASE, ALTER TABLESPACE, ..., and object privileges enabling the assigned user to access certain tables required for database administration actions performed with SAP tools (BR*TOOLS)
SAPCONN	The SAPCONN role is a specially defined database role, which contains all database authorizations required by the SAP application (ABAP and Java stack). It is independent of the SAP release.
.....	

Figure 21: Oracle Database Roles

Generally, any number of database roles can be created in an Oracle database. The SAP installation creates two additional roles, called SAPDBA and SAPCONN.

Based on object privileges, the role SAPDBA gives a user the ability to access certain tables used by SAP tools for database administration. Examples of such tables are DBCSTATC, SDBAD, DBSTATHORA, and so on. The SAPDBA role is assigned to the OPSS\$ user at installation. If the role assignment is accidentally deleted, the SAPDBA role can be reassigned to the user using the SQL script `sapdba_role.sql` as described in SAP Note 134592.

Up to and including Oracle release 10.1, the CONNECT role is assigned to the user SAP<Schema ID> or SAPR3. This Oracle standard role comprises a relatively large number of database authorizations. For security reasons, as of Release 10.2, Oracle has restricted the CONNECT role to the CREATE SESSION privilege. As of Release 10.2, only application-specific database roles (SAPDBA, SAPCONN) will be used for SAP database users.

When installing an SAP system with Oracle 10.2, the SAPCONN role is automatically assigned to the SAP database users. When you upgrade the database, the SAPCONN role must be created explicitly. The `sapconn_role.sql` script is used for this purpose with an installation description in the SAPEXE directory. You can also find this in the appendix to the SAP Note 834917.



Hint: The SAPCONN role can already be used with Oracle 9i as an option, and is required as of Oracle 10.2.

Connecting to Oracle

When establishing a connection to an Oracle instance, the user authentication used by Oracle depends on the type of connection request sent by the client.



Connection to Oracle and User Authentication

Connect request	Description
<code>connect <DB_user>/<PW>[@<DBSID>]</code>	Independent of operating system user; user authentication performed by Oracle
<code>connect /[@<DBSID>]</code>	Operating system authentication: OS user connected to the database as the corresponding OPS\$ user
<code>connect <DB_user>/<PW>[@<DBSID>] AS SYSDBA</code> or <code>connect /[@<DBSID>] AS SYSDBA</code>	Operating system authentication: OS user connected to the database as the user SYS with administrative privilege of SYSDBA; depends on membership in OS group (dba on UNIX)
<code>connect <DB_user>/<PW>[@<DBSID>] AS SYSOPER</code> or <code>connect /[@<DBSID>] AS SYSOPER</code>	Operating system authentication: OS user connected to the database with the schema PUBLIC with administrative privilege of SYSOPER; depends on membership in OS group (oper on UNIX)

The connect identifier `@<DBSID>` specifying the database ID is usually not necessary in an SAP environment where the DBSID is known from the environment variable `ORACLE_SID`.

Figure 22: Connection to Oracle and User Authentication

The keyword **CONNECT** is used within the Oracle client tool SQL*Plus for interactive connections to Oracle. The connect requests described here can, however, be used by other client applications, for example, by work processes of SAP instances through the database shared library for Oracle.

An operating system user can only perform `connect /` successfully if the corresponding OPS\$ user exists in the Oracle database.

An operating system user can only perform `connect / {AS SYSDBA | AS SYSOPER}` successfully if the user is member of the corresponding OS group (dba or oper on UNIX, ORA_DBA, ORA_<SID>_DBA or ORA_<SID>_OPER on Windows). Without this membership, the connection request is refused.

Connection AS SYSDBA gives you full Oracle instance and database administration privileges through the system privilege SYSDBA and the privileges assigned to user SYS, which is always effectively used in this case.

When you connect with AS SYSOPER, you are not authenticated as an explicit database user, but as the schema PUBLIC.

Connections with AS SYSDBA or AS SYSOPER replace CONNECT INTERNAL, which were used up to Oracle 8.x.

Security: SAP<SCHEMA-ID> Password

SAP work processes run on operating system level in the context of user <sapsid>adm (UNIX) or SAPService<SAPSID> (Windows). They must connect to Oracle with the username <SAPSCHEMA-ID> (or SAPR3 up to SAP Basis 4.6D). Because this Oracle user must be protected by a password, a mechanism has been implemented for work processes enabling them to find this password. This procedure is based on storing the password of user SAP<SCHEMA-ID> not only in an Oracle system table, but additionally in a special table called *SAPUSER*, which is created in the schema of user OPSS<SAPSID>ADM on UNIX, or of OPSS<DOMAIN>\<SAPSID>ADM on Windows. On Windows, this table is also accessible by the user OPSS<DOMAIN>\SAPService<SAPSID>.

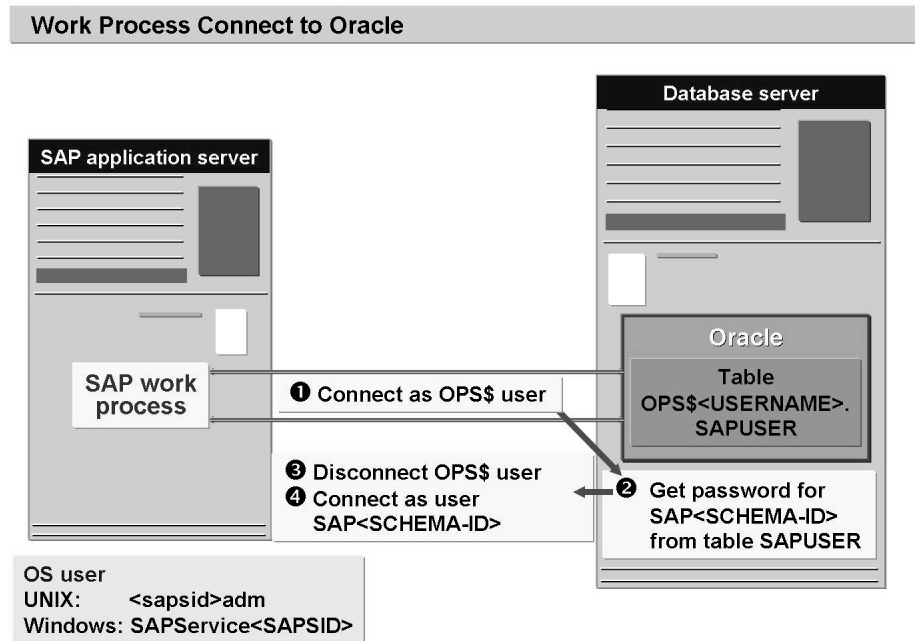


Figure 23: Work Process Connection to Oracle

When a work process starts and tries to connect to the Oracle database, the following happens:

1. The work process logs into the database as its corresponding OPSS\$ user with operating system authentication.
2. The work process sets a SELECT statement in the SAPUSER table and reads the password of SAP<SCHEMA-ID>.
3. The work process disconnects from Oracle.
4. The work process now connects with username SAP<SCHEMA-ID> and the password retrieved from the table SAPUSER.

If any of the steps cannot be processed successfully, the connection cannot be established and the work process stops with an error.



Caution: Never change the password of SAP<SCHEMA-ID> exclusively by Oracle methods. The password would not be modified in the table SAPUSER; consequently, work processes would send a connect request with the wrong password, which would be refused by Oracle. The SAP<SCHEMA-ID> password may only be changed with the SAP tool BRCONNECT.

The SAP<SCHEMA-ID> password is stored encrypted in the table SAPUSER.

NET Services

If an Oracle client, such as an SAP instance, is running on a computer other than the database server, SAP work processes and their dedicated shadow processes communicate over the network. As communication protocol, TCP/IP is used, and on top of it a software layer called OracleNet (Oracle Network Services). The work processes of an SAP instance configured on the database server use the interprocess communication (IPC) protocol Named Pipes to communicate with dedicated shadow processes running on the same server.

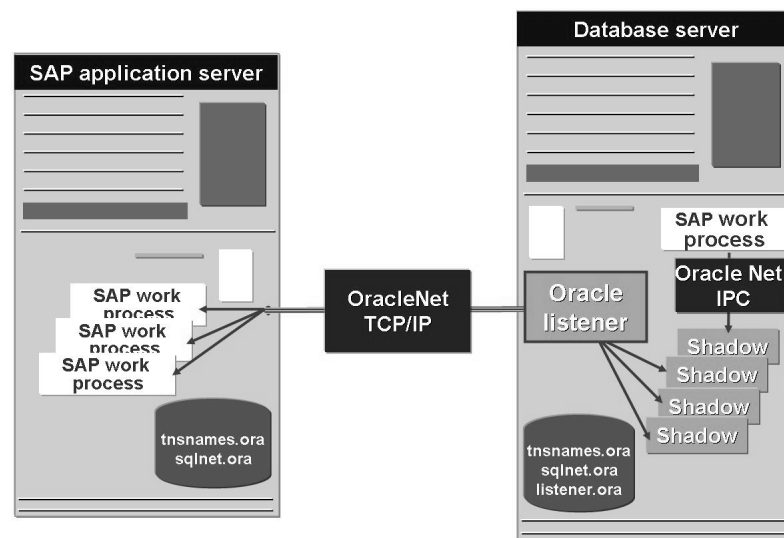


Figure 24: OracleNet Basics

OracleNet resides both on the client and on the Oracle database server. It is responsible for establishing and maintaining the connection between the client application and server, as well as for exchanging messages between them using standard protocols such as TCP/IP.

There is a special process called listener (more precisely, OracleNet listener) on the server whose responsibility is to listen for incoming connection requests. When the listener receives such a client request for a network session with the database server, the listener forwards the requests to the server, assuming that the client information matches the listener information. Once a connection is established, the client and Oracle database server communicate directly with one another.

The listener is configured with a protocol address. Only clients configured with the same protocol address can send connection requests to the listener. There are three operating system files used for this purpose, all of them stored in the directory `$ORACLE_HOME/network/admin`:

listener.ora

listener.ora configures the listener and, as such, is only used on the database host. It is read when the listener is started. The configuration information specified in this file determines OracleNet settings, such as the network protocol to be used, host name, port, and the default tracing information. listener.ora must contain all Oracle system IDs and protocol addresses for which the listener should accept connection requests.

tnsnames.ora

tnsnames.ora contains a list of service names for all databases that can be accessed in the network.

sqlnet.ora

sqlnet.ora can contain client side information, such as a client domain to append to unqualified service names or net service names, or optional diagnostic parameters used for client tracing and logging.

Oracle Listener

For OracleNet to accept connections on the database server, the listener must be running. The Oracle utility *lsnrctl* is used to start and stop the listener and to check the status of OracleNet connections. When listener is started:

- In a UNIX environment, the process *tnslsnr* is started.
- In a Windows environment, the service *Oracle<DBSID><Release>TNSListener* is started.

If several Oracle instances are installed on one host, there is usually just one listener process running on the host, serving all active Oracle instances.

The Oracle listener is controlled by the command line tool LSNRCTL.



```

LSNRCTL> help
The following operations are available
An asterisk (*) denotes a modifier or extended command:

→ start          → stop          → Status
services         → version       reload
save_config      → trace        change_password
quit            → exit         set*
show*

LSNRCTL> status
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=IPC)(KEY=EXTPROC)))
STATUS of the LISTENER
-----
Alias                LISTENER
Version              TNSLSNR for 64-bit Windows: Version 10.2.0.2.0 - Production
Start Date           16-NOV-2007 15:52:09
Uptime                9 days 18 hr. 27 min. 41 sec
Trace Level           off
Security              ON: Local OS Authentication
SNMP                 OFF
Listener Parameter File G:\ORACLE\DEV\102\network\admin\listener.ora
Listener Log File      G:\ORACLE\DEV\102\network\log\listener.log
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\EXTPROC1ipc)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=twdf9999)(PORT=1527)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\DEV.WORLDipc)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\T00.WORLDipc)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\DEVipc)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(PIPENAME=\\.\pipe\T00ipc)))
Services Summary...
Service "DEV" has 1 instance(s).
Instance "DEV", status UNKNOWN, has 1 handler(s) for this service...
...
The command completed successfully
LSNRCTL>

```

Figure 25: Managing the listener: LSNRCTL

To return a list of available commands, enter “help” if the *lsnrctl* command prompt appears.

Command `lsnrctl status` displays information such as OracleNet version, listener program start time, and the location of parameter and listener log files.

Database server listener tracing can be enabled by setting trace level information in the file `listener.ora` or by turning it on through the program LSNRCTL. Valid options for listener tracing are:

NOT SET

No tracing (default)

USER

Limited level of tracing information

ADMIN

Detailed trace



Caution: Use tracing for diagnostic purposes only. Do not leave tracing on indefinitely in a production system.

The listener can be pinged using the Oracle command TNSPING. To ping the listener, use `tnsping <DBSID>`. The result indicates if:

- the connection can be established.
- `<DBSID>` cannot be resolved in `tnsnames.ora`.
- The listener is not configured to communicate with `<DBSID>` through `listener.ora`.
- The listener is not running on the database server.

Exercise 2: Establish connection to the database

Exercise Objectives

After completing this exercise, you will be able to:

- Use different methods to connect to the database

Business Example

You need to connect to the database with SYSDBA privileges, but the connection does not work.

Task:

Connect to the database with different methods without specifying a password.

1. Call `sqlplus /nolog` to start SQLPLUS without automatically connecting to the database. Then use the `connect` command with different arguments to connect to the database. Does the connection work or not? Explain why.

First perform a `connect /`.

2. Use `connect /@<SID+1>` where `<SID+1>` is your SID plus one (for example, if your SID is T05, use `connect /@T06`).
3. Use `connect /@<SID+1> as sysdba` where `<SID+1>` is your SID plus one (for example, if your SID is T05, use `connect /@T06 as sysdba`).
4. **(Optional)** List the users registered in the database.

Solution 2: Establish connection to the database

Task:

Connect to the database with different methods without specifying a password.

1. Call `sqlplus /nolog` to start SQLPLUS without automatically connecting to the database. Then use the connect command with different arguments to connect to the database. Does the connection work or not? Explain why.

First perform a connect `/`.

a)

```
G:\oracle\T99>sqlplus /nolog
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 12:35:10 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
SQL> connect /
```

```
ERROR:
```

```
ORA-01034: ORACLE not available
```

```
ORA-27101: shared memory realm does not exist
```

```
SQL>
```

This is the OPSS\$ connect. It works because you are logged on to the operating system as user <sapsid>adm, and database user OPSS\$<DOMAIN>/<SAPSID>ADM exists.

The OPSS\$ user can only log onto a running database. The two ORA error messages indicate that the database is not running.

- b) Now start the database. Use `connect / as sysdba` to perform a connect with SYSDBA privileges. This connection works because the operating system user you are logged onto belongs to the operating system group ORA_<DBSID>_DBA.

Then start the database with the command: `startup`

```
SQL> connect / as sysdba
```

```
Connected to an idle instance.
```

```
SQL> startup
```

```
ORACLE instance started.
```

Continued on next page


```
Total System Global Area 134217728 bytes
Fixed Size                 2150000 bytes
Variable Size             113127824 bytes
Database Buffers          16777216 bytes
Redo Buffers               2162688 bytes
Database mounted.
Database opened.
SQL>
```

- c) Now perform a connect /.

```
SQL> connect /
Connected.
SQL>
```

After starting the database, you can now log on with connect /.

2. Use connect /@<SID+1> where <SID+1> is your SID plus one (for example, if your SID is T05, use connect /@T06).

- a)

```
SQL> connect /@T06
ERROR:
ORA-01017: invalid username/password; logon denied
```

```
Warning: You are no longer connected to ORACLE.
SQL>
```

- b) This is the OPS\$ connect again. This time it does not work, because on database T06 no user OPS\$<DOMAIN>/T05ADM exists; only OPS\$<DOMAIN>/T06ADM exists (given the example above).

Continued on next page

3. Use connect /@<SID+1> as sysdba where <SID+1> is your SID plus one (for example, if your SID is T05, use connect /@T06 as sysdba).

a)

```
SQL> connect /@T06 as sysdba
ERROR:
ORA-01031: insufficient privileges
```

- b) This time the connect does not work because on database T06, your operating system user t05adm does not belong to operating system group ORA_T06_DBA (given the example above).

4. **(Optional)** List the users registered in the database.

a)

```
SQL> connect /
Connected.
SQL> select username from dba_users;
```

```
USERNAME
-----
SYS
SYSTEM
OPS$TWDF9999\T99ADM
OPS$TWDF9999\SAPSERVICET99
SAPT99
...
```



Lesson Summary

You should now be able to:

- Name the default operating system and database users
- Explain Oracle communication over a network (NETServices)
- Describe the function of the Oracle listener
- Start and stop the Oracle listener

Lesson: Database Administration Tools

Lesson Overview

This lesson introduces the Oracle administration tool SQL*Plus. You will learn which SAP tools are available for database administration, and how they are used.



Lesson Objectives

After completing this lesson, you will be able to:

- Identify the main tools for administration of the Oracle database
- Describe basic functions of SQL*Plus
- Use SAP tools for administration of the Oracle database: BR*Tools
- Explain the usage of the SAP CCMS for the administration of Oracle databases

Business Example

To monitor and manage the database and to perform backups, a database administrator usually creates scripts to simplify the work. To perform these tasks on SAP systems using the Oracle database system, SAP delivers the BR*Tools to provide an easy interactive and batch interface. As a database administrator, you wish to learn more about these tools.

Overview: Database Administration Tools

With the Oracle tool SQL*Plus, you can:

- Start and stop the database
- Log into the database
- Perform database administration
- Enter SQL*Plus commands to configure an SQL*Plus environment
- Enter, edit, store, retrieve, and run SQL commands and SQL scripts
- Redirect the output of query results to text files

However, administration of an Oracle database in an SAP system can also be performed with a set of SAP administration tools for Oracle on operating system level. These are called **BR*Tools**. They are installed automatically on the database server in the directory `/usr/sap/<SID>/SYS/exe/run`.



Caution: Although the SAP administration tool SAPDBA is still available for Oracle 9i, SAP strongly recommends that you not use it any more because the functions of SAPDBA are no longer being developed. Oracle administration should now be performed exclusively with BR*Tools. BR*Tools can be used for all SAP releases providing that you are using an Oracle database release 9i or higher.

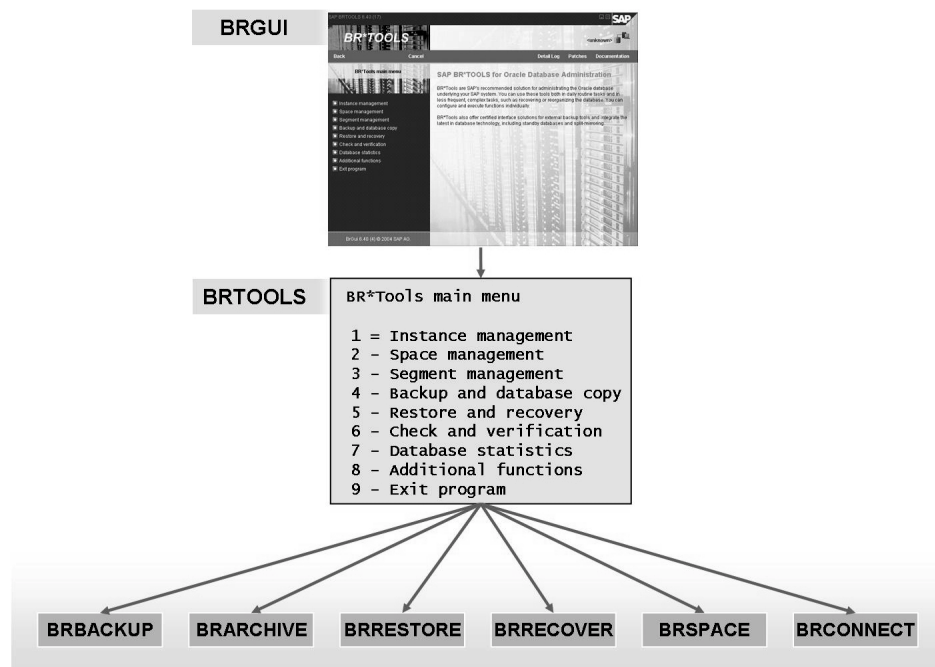


Figure 26: Hierarchy of SAP Tools for Oracle Administration

BRTOOLS is provided as a character-based user interface for the functional programs BRBACKUP, BRARCHIVE, BRRESTORE, BRRECOVER, BRSPACE, and BRCONNECT. While BRTOOLS enables the administrator to manage the database through menus using a simple Telnet connection or on a console, the graphical user interface of the BRGUI program offers more convenient database administration.

BRGUI is a Java -based graphical user interface for BRTOOLS. BRGUI itself does not offer any logic for database administration; it calls BRTOOLS, displays the BRTOOLS menus, and forwards selections and mouse clicks to BRTOOLS. Using a remote shell, it is possible to run BRGUI on the Administrator Workbench by calling BRTOOLS on the remote database server.

BR*Tools



Oracle Administration Tools on Operating System Level

Oracle tool

Name	Description
SQL*Plus	Interactive and batch query tool providing access to Oracle RDBMS through a command-line; installed with every Oracle server or client installation

SAP tools for Oracle administration (BR*Tools)

Name	Description
BRBACKUP	Backup of data files, control files, and online redo log files
BRARCHIVE	Backup of offline redo log files
BRRESTORE	Restore of data files, control files, and online and offline redo log files
BRRECOVER	Interactive parent tool for database restore and recovery
BRCONNECT	Database administration: database check, update of statistics, changing user password, and so on
BRSPACE	Database administration: instance management, space management, and reorganization
BRTOOLS	Interactive tool for calling the other tools through menus
BRGUI	Graphical user interface for BRTOOLS

Figure 27: Oracle Administration Tools

When the database administrator has to perform a certain administrative task, instead of calling a tool such as BRRESTORE, BRBACKUP, or BRCONNECT directly (which for some functions requires a specification of the corresponding function and options in detail), he or she can use the interactive, menu-driven program BRTOOLS. Through nested menus, this tool lets you choose the action and all the necessary options you need (including logon information). The diagram Menu Structure of BR*Tools provides an overview of the menu structure.

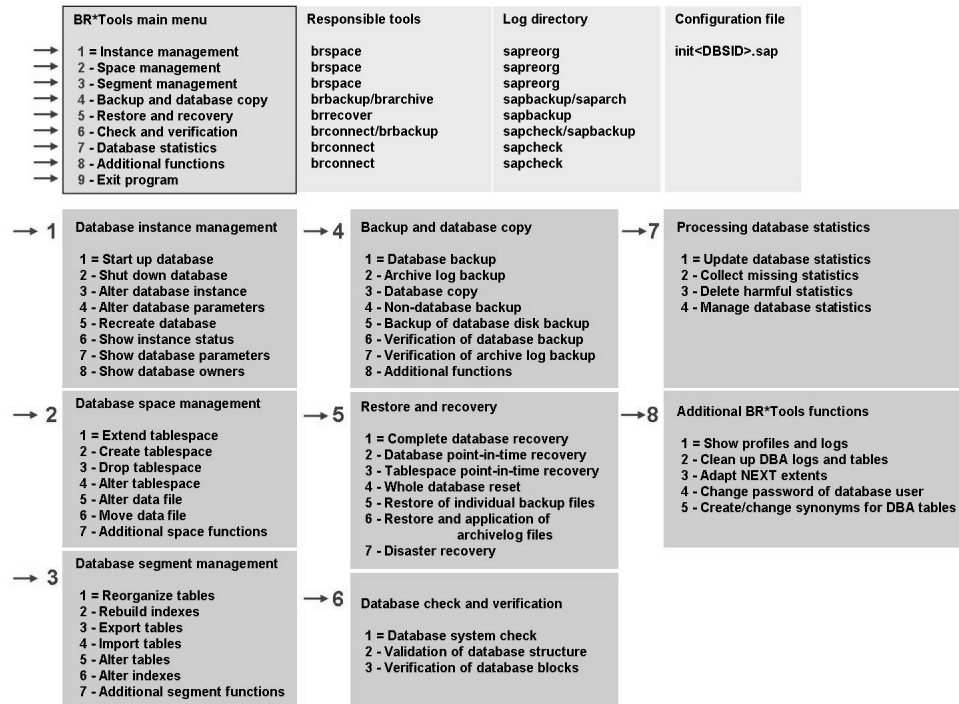


Figure 28: Menu Structure of BR*Tools



Hint: Do not mix up the two names BR*Tools and BRTOOLS.

- BR*Tools is the program package containing BRBACKUP, BRARCHIVE, BRRESTORE, BRRECOVER, BRSPACE, BRCONNECT, and BRTOOLS.
- BRTOOLS is the interactive program that displays menus from which the other BR* programs are called.



BR*Tools: Types of programs

Type	Description
Functional programs	BRBACKUP, BRARCHIVE, BRRESTORE, BRRECOVER, BRSPACE, and BRCONNECT. These programs perform actions on the database.
Help programs	BRTOOLS and BRCONNECT. Help programs are called by other programs; from BRTOOLS, all functional programs can be called interactively
Batch programs	BRBACKUP, BRARCHIVE, BRRESTORE, and BRCONNECT. These tools do not have own menus
Interactive programs	BRSPACE and BRRECOVER (besides BRTOOLS itself). These tools offer their own menus.

Figure 29: BR*Tools: Program Types

Within BR*Tools, we distinguish between **functional programs** and **help programs**, as well as **batch programs** and **interactive programs**:

functional programs

These are the programs performing administrative actions on the database. The functional programs are BRBACKUP, BRARCHIVE, BRRESTORE, BRRECOVER, BRSPACE, and BRCONNECT.

help programs

BRTOOLS and BRCONNECT are help programs. They help in two ways:

- From BRTOOLS, all functional programs can be called interactively. BRTOOLS offers menus to select the action, enter parameters and options, and then call the functional programs with the corresponding options and parameters.
- BRTOOLS and BRCONNECT are also called by other tools. They perform certain actions during backup and restore (BRTOOLS as a help tool for BRBACKUP, BRARCHIVE, and BRRESTORE) and monitor the database during a backup (BRCONNECT, internally called by BRBACKUP).



Hint: In previous releases, BRTOOLS and BRCONNECT were already delivered having only this internal help functionality. If you get an error message instead of a menu when calling BRTOOLS, you have a BR*Tools version older than 6.20 and you should get the newest BR*Tools.

batch programs

The tools BRBACKUP, BRARCHIVE, BRRESTORE, and BRCONNECT do not have own menus built in. Their functionality is selected exclusively by options. To use them interactively, call BRTOOLS and select actions, options and parameters from the menu provided by BRTOOLS.



Hint: For batch programs, **all** required options must be specified in BRTOOLS when directly called from the command line. If any required option is not specified, an error message is displayed.

interactive programs

The functional programs BRRECOVER and BRSPACE are (besides BRTOOLS itself) interactive tools. Their functionality is mainly menu driven.

You can also call BRRECOVER and BRSPACE directly from the command line:

- When no option is specified, the menu of the default function (BRSPACE: *Show database information*) and the default recovery type (BRRECOVER: *Complete database recovery*) is shown.
- When a function (BRSPACE) or recovery type (BRRECOVER) was specified on the command line, the corresponding menu is shown and further input is performed from the menus.
- When further options are specified but they are not complete, missing options and parameters must be selected from a menu.
- To force an interactive program to run in batch mode, call the interactive program with the option `-c force`. The interactive program will then start in batch mode. If mandatory options were not specified or any option or parameter is wrong, the tool will end with an error message.



Caution: Be careful, because the default values for all optional parameters are selected when you perform the corresponding action.

When calling any space management functionality of BRSPACE from BRTOOLS or BRGUI (*Space management* menu), after selecting a function (for example *Extend tablespace*), you can choose either the **main menu mode** or the **quick mode**:



main menu mode

If you do not enter anything in the menu called BRSPACE options for `<function>` (except specifying another BRSPACE profile and/or username and password if required) and just press *Continue*, BRSPACE will display the main menu of the corresponding function and all required options and parameters can be selected from BRSPACE menus.

quick mode

If you enter further information in the BRSPACE options for <function> menu, first of all the database object(s) the selected function is to be performed on, BRSPACE will append further command line options to the command line starting BRSPACE. In this case, BRSPACE will skip the main menu of the corresponding function and (if one object was specified in the BRTOOLS menu) also the object selection menu. You can then directly enter function options in the input menu.



Hint: Use the quick mode only if you already know the objects for which you want to perform the function. If you want to select objects from a list, or see a list of possible entries for parameters or options, use the main menu mode. Not all options of BRSPACE and BRRECOVER can be set from BRTOOLS, but these specific options can be set in menus provided by the interactive tool itself. When calling an interactive tool in batch, all options can be set on the command line.



Caution: Before executing an action, BRTOOLS will display the complete command line of the batch program it will call. You can use this command line as a reference to call this command later in batch, but be aware that changing the command line might end in errors.

BRSPACE displays the SQL command it will execute before performing the operation. Only use this command or a changed version if you are an Oracle expert, because:

- In many cases, BRSPACE will perform additional actions before and after executing the SQL command (so do not enter the command directly in SQL*Plus).
- BRSPACE will perform several checks before creating the SQL command shown. If you change the command, you take full responsibility for the action performed by the changed command.

Common Command Line Options of BR*Tools



Common Command Line Options of BR*Tools

Tool	Option	Meaning
all	-h -help	List of all possible possible options and functions.
BRSPACE, BRCONNECT	-h -help <function>	Lists all possible options of <function>
all	-c -confirm	Unattended Mode - no confirmation required. Standard: attended mode
BRSPACE, BRRECOVER	-c -confirm force	Interactive tools stop at menus even with -c confirm. To run them in batch, use option force.
all	-u -user [<name> [/<pw>]]	User name and password for database connection

If you run any tool of BR*Tools with the option -h | -help, you get a list of all possible options. For BRSPACE and BRCONNECT, this includes a list of all program functions:

If you need just a listing of all options for a particular function of BRSPACE or BRCONNECT, call the tool with -h | -help and the corresponding function name, for example, `brspace -h tscreate`.

All tools run per default in the attended mode, which means that each single step must be confirmed by the user. To avoid this, start a tool with the option -c | -confirm. The interactive tools BRRECOVER and BRSPACE still stop at menus even if started with -c | -confirm. To run them in a batch, use the option -c | -confirm force, which suppresses all confirmation messages and accepts the default input value in menus.

To avoid passwords having to be entered interactively or appear on the command line, BR*Tools normally use the OPS\$ user to connect to the database. To use a different user for the database connect you can specify the option `-u | -user`.



Hint: BRSPACE and BRRECOVER always make a CONNECT / AS SYSDBA because their actions require SYSDBA privilege. The username/password specified with the option `-u | -user` is only used for other BR*Tools called by BRSPACE or BRRECOVER, or for BRBACKUP, BRARCHIVE, and BRCONNECT called through BRTOOLS menus. Therefore, BRSPACE and BRRECOVER must be called from an operating system user belonging to the DBA group (<sapsid>adm on Windows, ora<dbsid> on UNIX) on the database server.

Working with BRTOOLS

After you have started BRTOOLS, you must first choose a type of administrative activity from the main menu.



```
G:\oracle\T99>brtools
BR0651I BRTOOLS 7.00 (30)

BR0280I BRTOOLS time stamp: 2007-11-26 11.10.52
BR0656I [Choice menu 1] please make a selection
-----
BR*Tools main menu

1 = Instance management
2 = Space management
3 = Segment management
4 = Backup and database copy
5 = Restore and recovery
6 = Check and verification
7 = Database statistics
8 = Additional functions
9 = Exit program

Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
BR0662I Enter your choice:
1
BR0280I BRTOOLS time stamp: 2007-11-26 11.11.12
BR0663I Your choice: '1'

BR0280I BRTOOLS time stamp: 2007-11-26 11.11.12
BR0656I [Choice menu 3] please make a selection
-----
Database instance management

1 = Start up database
2 = Shut down database
3 = Alter database instance
4 = Alter database parameters
5 = Recreate database
6 = Show instance status
7 = Show database parameters
8 = Show database owners
9 = Reset program status

Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
BR0662I Enter your choice:
```

Figure 30: Working with BRTOOLS: Menu Structure

The main menu and all menus on the next level are called choice menus; you can make an independent choice from the menu in any sequence, and you can repeat the choice as often as required. Other types of menus are the control menu (the steps

presented by the menu must be processed in the given sequence, for example during restore and recovery), input menu (values of required parameters or options are suggested and can be modified), and list menu (several items are listed from which you can select one or more entries, for example, one database backup from several available backups during restore).

The so-called standard keys, used in all types of menus, have the following meanings:

c - cont

continues to the next menu or program step

b - back

goes back to the previous menu or program step



Hint: When you enter BRSPACE from BRTOOLS using quick mode, going back deactivates the quick mode and you can reach the main menu of the corresponding function.

s - stop

cancels the active program

r - refresh

refreshes the screen and makes some plausibility checks

h - help

calls context-specific help

Altering a Database User's Password

For security reasons, the passwords of the standard database users and of the SAP user SAPR3 or SAP<SCHEMA-ID> should be changed regularly.

As mentioned earlier, the password of database user SAP<SCHEMA-ID> (or SAPR3) should not be changed with Oracle methods because it must be maintained in the *SAPUSER* table. The function *chpass* of BRCONNECT should be used instead; it changes the password both in the Oracle system table and in *SAPUSER*.

To change the password of a database user, call BRTOOLS or BRGUI and choose *Additional functions* → *Change password of SAP user*.



BRCONNECT: Changing Password of SAP Owner

```
G:\oracle\T99>brtools  
BR0651I BRTOOLS 7.00 (30)
```

BR0280I BRTOOLS time stamp: 2007-11-26 11.47.16

BR0656I Choice menu 1 - please make a selection

BR*Tools main menu

- 1 = Instance management
- 2 - Space management
- 3 - Segment management
- 4 - Backup and database copy
- 5 - Restore and recovery
- 6 - Check and verification
- 7 - Database statistics
- 8 - Additional functions
- 9 - Exit program

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

8

BR0280I BRTOOLS time stamp: 2007-11-26 11.47.20

BR0663I Your choice: '8'

BR0280I BRTOOLS time stamp: 2007-11-26 11.47.20

BR0656I Choice menu 2 - please make a selection

Additional BR*Tools functions

- 1 = Show profiles and logs
- 2 - Clean up DBA logs and tables
- 3 - Adapt NEXT extents
- 4 - Change password of database user
- 5 - Create/change synonyms for DBA tables
- 6 - Reset program status

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

4

BR0280I BRTOOLS time stamp: 2007-11-26 11.47.25

BR0663I Your choice: '4'

BR0280I BRTOOLS time stamp: 2007-11-26 11.47.25

BR0657I Input menu 59 - please check/enter input values

```
-----
BRCONNECT options for changing password of database user
```

- ```

1 - BRCONNECT profile (profile) [initT99.sap]
2 - Database user/password (user) [/]
3 ~ Database owner to change password (owner) . []
4 - Message language (language) [E]
5 - BRCONNECT command line (command) [-p initT99.sap -l E -f chpass]
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help

```

```
BR0662I Enter your choice:
```

```
3
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.47.35
```

```
BR0663I Your choice: '3'
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.47.35
```

```
BR0681I Enter string value for "owner" []:
```

```
SAPT99
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.47.39
```

```
BR0683I New value for "owner": 'SAPT99'
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.47.39
```

```
BR0657I Input menu 59 - please check/enter input values

```

```
BRCONNECT options for changing password of database user
```

- ```

1 - BRCONNECT profile (profile) ..... [initT99.sap]
2 - Database user/password (user) ..... [/]
3 ~ Database owner to change password (owner) . [SAPT99]
4 - Message language (language) ..... [E]
5 - BRCONNECT command line (command) ..... [-p initT99.sap -l E -f chpass
-o SAPT99]
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
```

```
BR0662I Enter your choice:
```

```
c
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.48.00
```

```
BR0663I Your choice: 'c'
```

```
BR0259I Program execution will be continued...
```



```
BR0291I BRCONNECT will be started with options '-p initT99.sap -l E -f chpass -o
SAPT99'
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.48.00
```

```
BR0670I Enter 'c[ont]' to continue, 'b[ack]' to go back, 's[top]' to abort:
c
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.48.02
```

```
BR0257I Your reply: 'c'
```

```
BR0259I Program execution will be continued...
```

```
#####
```

```
BR0801I BRCONNECT 7.00 (30)
```

```
BR0280I BRCONNECT time stamp: 2007-11-26 11.48.03
```

```
BR0263I Enter password for database user 'SAPT99' (maximum 30 characters):
```

```
BR0280I BRCONNECT time stamp: 2007-11-26 11.48.12
```

```
BR0263I Reenter password for database user 'SAPT99' (maximum 30 characters):
```

```
BR0280I BRCONNECT time stamp: 2007-11-26 11.48.14
```

```
BR0829I Password changed successfully in database for user SAPT99
```

```
BR0830I Password changed successfully in table OPS$T99ADM.SAPUSER for user
SAPT99
```

```
BR0280I BRCONNECT time stamp: 2007-11-26 11.48.14
```

```
BR0802I BRCONNECT completed successfully
```

```
#####
```

```
BR0292I Execution of BRCONNECT finished with return code 0
```

```
BR0280I BRTOOLS time stamp: 2007-11-26 11.48.14
```

The option `-u` | `-user` defines the user name and password used by the SAP tool to log on to the database. This user must be defined in the database and have at least SYSOPER authorization.

The option `-o` | `-owner` specifies the user for which the password should be changed. You can change the password of any database user using this function.

Appendix: Menu Symbols in BR*Tools

Special symbols are used in BR*Tools menus at the beginning of every line. Remembering their meaning helps the administrators to orientate themselves better in the menus and to use the tools more effectively.

Menu Symbols in BR*Tools



Character Interface	Used in Menu Types	Meaning
+	Control Choice	The action is completed
Information message	Display	• An error message is written to the detail log in the form: BRxxxxl. • used for all rows in the display menu
Warning message	Not used	An error message is written to the detail log in the form: BRxxxxW
Error message	not used	An error message is written to the detail log in the form: BRxxxxE
–	Control Choice List	You can choose or execute this now
*	Control Input	• You cannot choose or execute this now • Display entry, no input possible

Menu Symbols in BR*Tools (2)



Character Interface	Used in Menu Types	Meaning
–	Input	You can change this parameter
~	Input	You can change this optional parameter or reset its value to null (use a single space for this in the character interface)
stop	All menus	This cancels the active program
help	All menus	This calls context-specific help

Menu Symbols in BR*Tools (3)



Character Interface	Used in Menu Types	Meaning
back	All menus	This goes back to the previous menu or program step
continue yes	All menus	This continues to the next menu or program step
no	not used	This skips the following actions to the next normal program step; it is recorded as BR0676I in the detail log
=	Control Choice List	This is the initial default choice, with a yellow background; it is automatically selected if you choose <i>Continue</i>
?	Input	You must enter a value for this parameter
#	Control Input	You cannot: <ul style="list-style-type: none"> • Execute this action • Change this parameter

Appendix: SQL*Plus

You start SQL*Plus by calling the program SQLPLUS. You are then immediately asked to enter a username and password to connect to your default database.



SQL*Plus: Connect

```
G:\oracle\T99>sqlplus
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 11:53:56 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
Enter user-name: system
```

```
Enter password:<password not shown on screen>
```

```
Connected to:
```

```
Oracle Database 10g Enterprise Edition Release 10.2.0.2.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options
```

SQL>

If you don't want to enter the username/password interactively, you can:

- Enter the username/password as the first parameter when calling SQLPLUS, for example `sqlplus system/manager`.
- Call SQLPLUS with the option `/nolog`. In this case, the first SQL*Plus command must be a `connect username/password` command. Use this to call SQL scripts downloaded from the SAP Service Marketplace, which make its own connection to the database. The following example shows how the interactive logon is suppressed and the `CONNECT` command is necessary to enter SQL*Plus commands:



SQL*Plus: Commands

```
G:\oracle\T99>sqlplus /nolog
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 11:55:34 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
SQL> archive log list;
ORA-03114: not connected to ORACLE
SQL> connect / as sysdba
Connected.
SQL> archive log list;

Database log mode                No Archive Mode
Automatic archival                Disabled
Archive destination               G:\oracle\T99\oraarch\T99arch
Oldest online log sequence        51
Current log sequence              54
SQL>
```

The general syntax to specify username and password on the connection is `username/password[@DBSID]`. SQL*Plus, by default, connects to the database defined in the environment variable `%ORACLE_HOME%` (Windows) or `$ORACLE_HOME` (UNIX). To connect to another database, enter the `<SID>` to connect to directly after the password without a blank, delimited by the *at* sign (`@`).

To connect to the database with a user having the `SYSDBA` privilege, but without specifying a password, do a `CONNECT / AS SYSDBA`.

Customizing SQL*Plus

SQL*Plus maintains system variables to allow you to set up a particular environment for an SQL*Plus session. You can change these system variables with the SET command:

```
SET <VARIABLE> <VALUE>
```

You can list system variables with the SHOW command:

```
SHOW <VARIABLE>
```

```
SHOW ALL
```

The following table shows the most commonly used SQL*Plus system variables. For a complete list, refer to Oracle documentation.



SQL*Plus: System Variables

Variable	Value	Description
PAGESIZE	integer	Number of lines per page (after this number of lines, a new header line is displayed)
PAUSE	ON OFF	If ON, output is stopped after PAGESIZE lines; continue with <i>Return</i> .
HEADING	ON OFF	Enabling / disabling output of column headings
LINESIZE	integer	Display width for data; after this number of characters, the output line is wrapped or truncated, depending on the value of WRAP
WRAP	ON OFF	If ON, lines are wrapped after LINESIZE characters; if OFF, lines are truncated

Variable	Value	Description
SPOOL	<filename> OFF	Copies the output of all subsequent commands and their output to <filename>, until set OFF
AUTO [COMMIT]	ON OFF	If ON, pending changes are automatically committed to the database after each successful INSERT, UPDATE, or DELETE statement; if OFF, such changes must be committed manually
TERMOUT	ON OFF	Value OFF suppresses terminal output (only works in scripts)



Hint: If AUTOCOMMIT is OFF (the default setting), data modifications performed from SQL*Plus are committed in the database only when you either execute the command COMMIT in the same session, or you disconnect from the database.

Although SQL*Plus variables can be changed dynamically in an SQL*Plus session, a commonly used method is to create a script file containing a SET command for each variable you want to set, and to execute the file every time you start SQL*Plus:

- Create an ASCII file with the extension sql, for example, sqlplusenv.sql, in a directory from which you start SQL*Plus, for example in your home directory.
- Enter a SET command for each relevant variable in this script file, and save the contents.
- When starting SQL*Plus, enter the script file name as the last option of the sqlplus command, separated by a space, and with the @ sign in front of the file name (you may leave out the extension sql): sqlplus {username [/password] [/]} [@<DBSID>] @<script>

SAP Database Monitors

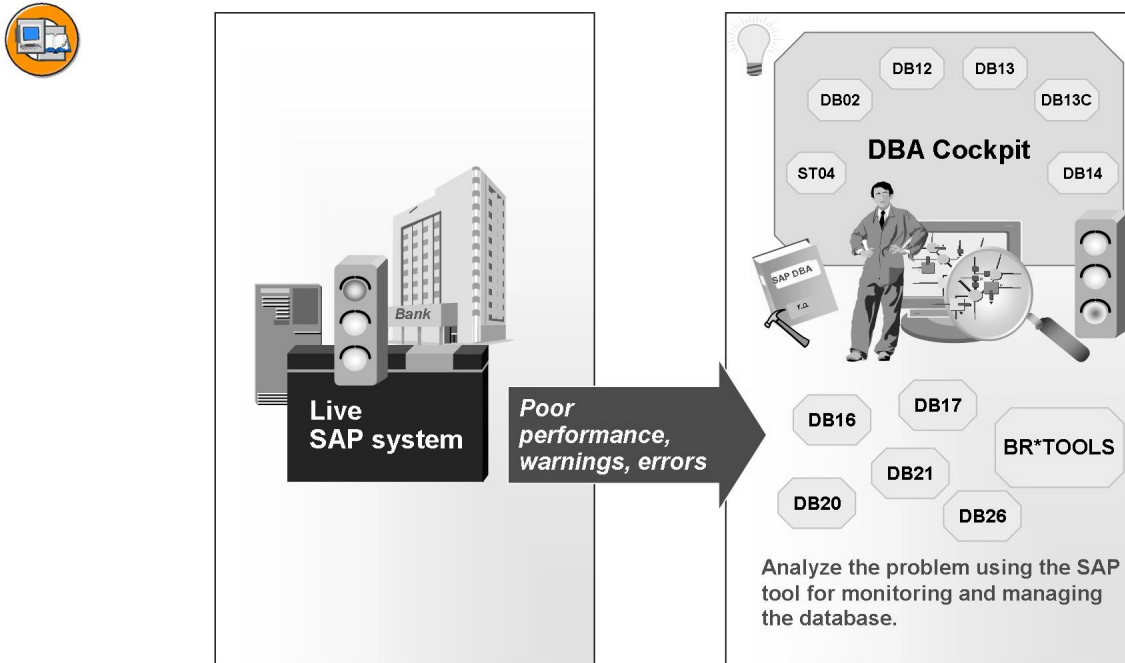


Figure 31: Database Administration and Monitoring in SAP System

To minimize system downtime and improve performance, you must schedule regular administrative jobs such as backups and database checks, and monitor your database daily. The DBA Cockpit, first delivered with SAP Basis Release 7.00 Support Package 12, forms the central access point for the monitoring and administration of SAP systems with an Oracle database.

The DBA Cockpit replaces various transactions that were previously used for monitoring and administration (see SAP Note 1028624). These include the following transactions that now lead to individual functions in the DBA Cockpit:

- The **DBA Planning Calendar** and the **central DBA planning calendar** (transaction DB13 and DB13C) is available for scheduling backups and other administrative jobs in your database system. You can use this transaction to schedule backups and administrative activities locally or centrally for several SAP systems and databases.
- You can use the **Backup Log Overview** (transaction DB12) to display the results of your backups and the status of the archive directory. If all backups are available for a restore and recovery, it also contains a function for checking the restore reports.
- The **DBA Operations Monitor** (transaction DB14) checks the status and logs of all database operations, including backup monitoring, updates of the optimizer statistics, and database checks.
- The **Database Performance Monitor** (transaction ST04) displays the most important indicators for Oracle database performance – such as buffer cache quality, statistics of user calls or number of block reads per SQL statement – and the configuration of the database management system and the database, either directly through showing current parameter values or with help of V\$ views.
- The **Tables and Indexes Monitor** (transaction DB02) monitors the storage behavior of the database (for example, space statistics showing the history of the database, or size and free space in each tablespace) and the status of the database objects (for example size of each table in kilobytes and blocks, or indexes that are defined in the ABAP Dictionary but missing in the database).

The above transactions are still available, but the transaction codes have been renamed to <PREVIOUS TRANSACTION CODE>OLD. This means that the transaction codes are now ST04OLD, DB02OLD, DB12OLD, DB13OLD, DB14OLD, and DB13COLD. The DBA Cockpit is called using transaction DBACOCKPIT.

In addition to the DBA Cockpit, we will also discuss the following transactions for monitoring the database:

- The **overview of database checks** (transaction DB16).
- Transaction DB17 to **view and maintain check conditions** used by a database system check.
- Transactions DB20 (**Edit table statistics**) and DB21 (**configuration of statistics**).
- The **database parameter overview** with history (transaction DB26).
- The **Database Alert Monitor** (can be started from transaction RZ20) monitors all preset alerts for different areas of the database.

These transactions are not discussed here in detail, but in the unit in which they are also required.

DBA Cockpit

The DBA Cockpit offers a navigation area, which is visible in all functions of the DBA Cockpit. This area contains a menu tree with the following access points:

- **Performance** (corresponds to the old transaction ST04)
- **Space** (corresponds to the old transaction DB02)
- **Jobs** (corresponds to the old transactions DB13, DB12, DB14, DB13C)
- **Diagnosis**

The individual functions of the DBA Cockpit occur under these four access points, which can each be called by double-clicking them.

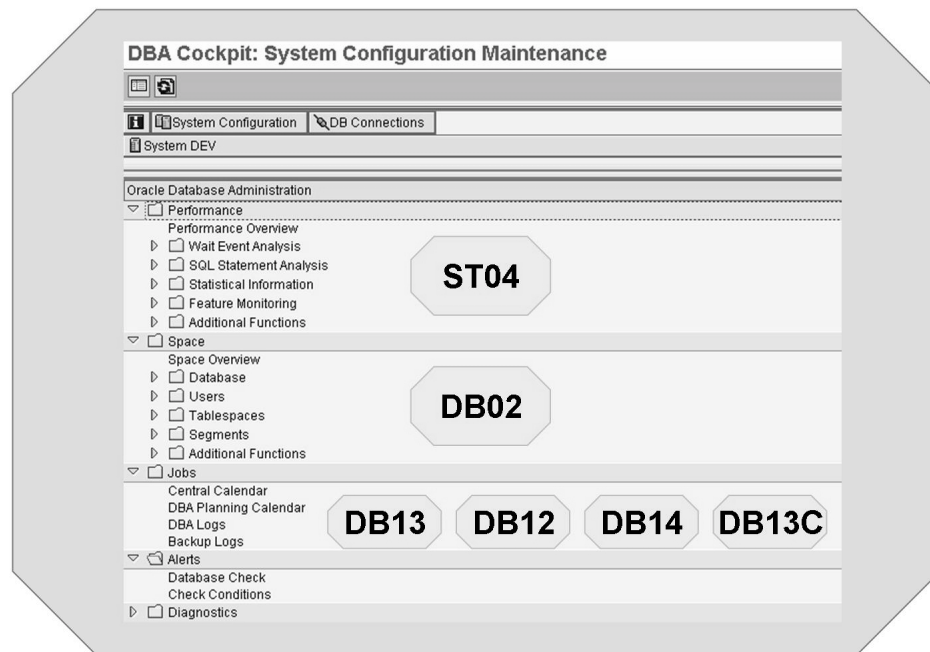


Figure 32: DBA Cockpit

The following prerequisites are required for monitoring and administration of the local system (that is, the system running on the DBA Cockpit):

- Specific database objects are required for some performance monitors in the DBA cockpit. These objects are created with an SQL script. The script and other information can be found in SAP Note 706927.
- For the new planning calendar, install BR*Tools 700, patch level 24 or higher.
- Some functions in the DBA Cockpit require the Oracle Active Workload Repository. This Oracle option must therefore be available (see SAP Note 1028068).

In addition, other corrections (see SAP Note 1028624) must be imported for DBA Cockpit with Basis 7.00 Support Package 12 and 13, which are first fixed in Support Package 13 or 14.

Administration and Monitoring of External Databases

One feature of the DBA Cockpit is the option of monitoring and administering external databases, including both ABAP and non-ABAP systems.

Not only Oracle databases can be connected as external databases, but also all other SAP supported database systems (Oracle, IBM DB2 UDB, IBM DB2 z/Os, IBM DB2 AS/400, MS SQL Server, MaxDB). To connect a non-Oracle database, the corresponding database client software and the SAP shared library must be installed.



Hint: If the system on which the DBA Cockpit is running is not an Oracle database system, and you want to connect external Oracle databases, then you must of course install the Oracle database client software and the corresponding SAP database shared library beforehand.

External Oracle databases are supported as of Version 9.2.



Caution: If you intend to connect external Unicode databases, the system on which the DBA Cockpit is running must be a Unicode system.

A functional secondary database connection is required to connect external databases (in other words, there must be an entry in the DBCON table). When adding an external database in the system configuration of the DBA Cockpit, you can create your own DBCON entry. To connect external Oracle databases, please note the other following prerequisites:

- The script contained in SAP Note 706927 must be executed on the external Oracle database.
- For the planning calendar (DB13) and the corresponding transaction DB12 and DB14, you must configure either an RFC ABAP connection (only possible for external ABAP systems) or a connection through the SAP Gateway or remote shell. For more information on this procedure, see SAP Note 1025707.
- In addition, a minimum of BR*Tools 700 patch level 24 (Oracle client 10) or BR*Tools 640 patch level 52 (Oracle client 9) must be installed on the external system.

DBA-Cockpit: Jobs

Local planning calendar

Transaction DB13 or *Jobs* → *DBA Planning Calendar* in the DBA Cockpit menu tree

From the DBA Planning Calendar, you can schedule periodic administrative jobs for a database, such as the database check, database backups, updating the optimizer statistics, and adapting next extents (pertinent only for dictionary managed tablespaces).

The previous planning calendar was replaced with the new one to comply with legal accessibility requirements in interface programs. The previous planning calendar is still available using the transaction DB13OLD. Completed actions can also be viewed in the old and new planning calendars. Actions that were scheduled in the old planning calendar also appear as scheduled in the new one. The reverse also applies; that is, actions that have been scheduled in the new planning calendar can also be viewed in the old one, except: Periodically scheduled actions in the new planning calendar with any repeat period other than weekly (for example, hourly or daily) do not appear in the old planning calendar. This can cause problems if you are using the old and new planning calendars at the same time. We therefore recommend that you do not use both versions at the same time. Even if the old calendar can still be used, it now has the status "deprecated". If you do not want to use the new planning calendar immediately, we still therefore recommend that you switch to it in the medium term. For more information on the new planning calendar (particularly the configuration and administration of external databases), see SAP Note 1025707 and the documentation.



DBA Cockpit: Jobs → DBA Planning Calendar (DB13)

Jobs: DBA Planning Calendar

Refresh Day Week Month Save Settings ... Legend ...

DBA Planning Calendar

System: DEV

Category: DBA Actions

CalendarID:

March 2008

Week	Monday	Tuesday	Wednesday
200810	March, 03	March, 04	March, 05

Action Pad

- Whole database offline + redo log backup
- Whole database online backup
- Whole database offline backup

Action Description

Whole database online backup

Planned Start: 07.03.2008 15:00:00

Status:

Action Parameters **Recurrence**

Recurrence Pattern

Every 1 at

☒ Day(s) at

☐ 00:00 ☐ 01:00 ☐ 02:00 ☐ 03:00 ☐ 04:00 ☐ 05:00
☐ 06:00 ☐ 07:00 ☐ 08:00 ☐ 09:00 ☐ 10:00 ☐ 11:00
☐ 12:00 ☐ 13:00 ☐ 14:00 ☒ 15:00 ☐ 16:00 ☐ 17:00
☐ 18:00 ☐ 19:00 ☐ 20:00 ☐ 21:00 ☐ 22:00 ☐ 23:00

☐ Hour(s) on ☐ Mon ☐ Tue ☐ Wed ☐ Thu ☐ Fri ☐ Sat ☐ Sun

☐ Week(s) on

☐ Once only

Recurrence Range

Start: 07.03.2008 at 15:00:00 ☒ No end date

☐ End after 1 Occurrences

☐ End by at 00:00:00

Execute action every 1 day(s) at 15:00

Figure 33: DBA Planning Calendar

The DBA planning calendar is a simple interface to schedule background jobs named *DBA:** performing administrative jobs. These background jobs look up table *SDBAC* to check, depending on the SAP release, the database system, and the operating system, which operating system level command should be executed on which server.

The DBA planning calendar provides templates for all activities that are recommended to be performed regularly on the database.



Caution: On SAP systems up to release 4.6C, table *SDBAC* contains entries to start *sapdba* to perform administrative jobs. As *sapdba* is no longer developed and its functionality is completely replaced by BR*Tools, SAP recommends updating table *SDBAC* to run BR*Tools instead of *sapdba* on SAP systems as of release 4.0B and database Oracle 8.1.7. SAP Note 403704 contains detailed instructions on how to perform this update.

Central Planning Calendar

Transaction DB13C or *Jobs → Central Planning Calendar* in the DBA Cockpit menu tree

The new central planning calendar is fundamentally different from the old one (which is still available as transaction DB13COLD) in terms of the configuration of the systems to be monitored. The new central planning calendar uses the system configuration of the DBA Cockpit to add or remove systems. To migrate the systems of the old central planning calendar to the new one, a menu option is available in the new central planning calendar under *Administration* → *Migration DB13C Configuration*.

DBA Operations Monitor

Use the DBA Operations Monitor (transaction DB14 or *Jobs* → *DBA Logs* in the DBA Cockpit menu tree) to monitor online database operations. You can also monitor the runtime of operations.



DBA Cockpit: Jobs → DBA Logs (DB14)

DBA Logs

DB Name: DEV Started: 15.02.2008
 DB Server: TWDF1902 Time: 12:55:55
 DB Release: 10.2.0.2.0

DBA

BRSPACE BRSPACE Operations
 BRCONNECT BRCONNECT Operations

Backups

BRBACKUP Database Backups
 BRARCHIVE Redo Log Backups

Others

Others Other Operations
 Data Archiving Non-SAP Data Archiving

All

All All Operations
 Function IDs All Function IDs:

Figure 34: DBA Operations Monitor

The DBA Operations Monitor provides an overview of activities of any of the BR*Tools.

To display specific database operations (for example, backup operations), choose the corresponding button. For your daily check you should choose *All* to see a list of all activities and their results. The colors indicate if an activity had warnings (color: yellow return code 0001) or errors (color: red, return code larger than 0001).



Hint: To restrict the activity list to a specific period, or to display only the activities with warnings or errors, first select the list of corresponding operations, and then select the *selection criteria* button.

To see the action log of operations, double-click the corresponding line. From here, you can view the detail log using the *Detail Log* button.

Transaction DB12 or *Jobs* → *Backup Logs* in the DBA Cockpit menu tree shows an overview of the backup logs. This shows the results of the data backups and the status of the archive directory. If all backups are available for a restore and recovery, it also contains a function for checking the restore reports.

DBA-Cockpit: Space

Transaction DB02 or *Space* → *Space Overview* in the DBA Cockpit menu tree shows the functions for monitoring disk space in the database.

This overview contains information about how much disk space the database is using. To ensure that the required data for this overview is available, a background job must be scheduled first.

Available disk space information about individual tablespaces or tables is provided in additional sub-monitors. Information is also provided about the growth of the individual database objects.

Mapping

These disk space monitoring functions are discussed in more detail in the unit *Monitors and Tools*.

DBA-Cockpit: Performance

You can analyze database performance with transaction ST04 or *Performance* → *Performance Overview* in the DBA Cockpit menu tree.

This course does not discuss performance analysis in more detail; it is discussed in the training module: “Database Administration (Oracle) II”.

Mapping

Exercise 3: Database Administration Tools

Exercise Objectives

After completing this exercise, you will be able to:

- Use BR*Tools

Business Example

You want to learn more about how BR*Tools are used to display information about the database, and perform some actions on the database.

Task:

Use BR*Tools. It is recommended to perform the exercises using BRGUI, but BRTOOLS can be used directly as well. For better readability, the solutions given come from BRTOOLS.

1. Use BR*Tools to display all tablespaces of your database.
2. Use BR*Tools to change the password of user SAP<SCHEMA-ID> to “secret”. Then change the password again to “sap”.



Caution: It is important for the exercises in the next units that the password of user SAP<SCHEMA-ID> is “sap”. If the password is different, scripts provided for the exercises will not work!

Solution 3: Database Administration Tools

Task:

Use BR*Tools. It is recommended to perform the exercises using BRGUI, but BRTOOLS can be used directly as well. For better readability, the solutions given come from BRTOOLS.

1. Use BR*Tools to display all tablespaces of your database.

- a) Call BRGUI or BRTOOLS and choose *Space management* → *Additional space functions* → *Show tablespaces*. Confirm your entries twice with “continue”.

- b)

List of database tablespaces

Pos.	Tablespace	Type	Status	ExtMan.	SegMan.	Backup	Files/AuExt.
	Total [KB]	Used [%]	Free [KB]	MaxSize [KB]	ExtSize [KB]		FreeExt.
	Largest [KB]						
1 -	PSAPT99	DATA	ONLINE	LOCAL	AUTO	NO	1/1
	20480	16.88	17024	51200	30720		1
	30720+:17024:0:0:0						
2 -	PSAPT99USR	DATA	ONLINE	LOCAL	AUTO	NO	2/0
	20480	0.63	20352	20480	0		2
	10176:10176:0:0:0						
3 -	PSAPTEMP	TEMP	ONLINE	LOCAL	MANUAL	NO	1/0
	20480	0.00	20480	20480	0		0
	0:0:0:0:0						
4 -	PSAPUNDO	UNDO	ONLINE	LOCAL	MANUAL	NO	1/0
	20480	0.31	20416	20480	0		229
	1024:1024:1024:1024:1024						
5 -	SYSAUX	DATA	ONLINE	LOCAL	AUTO	NO	1/0
	204800	26.66	150208	204800	0		1
	150208:0:0:0:0						
6 -	SYSTEM	DATA	ONLINE	DICT	MANUAL	NO	1/0
	409600	44.41	227704	409600	0		1
	227704:0:0:0:0						

2. Use BR*Tools to change the password of user SAP<SCHEMA-ID> to “secret”. Then change the password again to “sap”.

Continued on next page



Caution: It is important for the exercises in the next units that the password of user SAP<SCHEMA-ID> is “sap”. If the password is different, scripts provided for the exercises will not work!

- a) Call BRGUI or BRTOOLS and choose *Additional functions* → *Change password of SAP user*. If you select BRCONNECT options for changing password of database user in the next input menu

If no user name is entered in the option owner, the password for the SAP<SCHEMA-ID> user is changed.

- b) For details see the figure in the lesson.



Lesson Summary

You should now be able to:

- Identify the main tools for administration of the Oracle database
- Describe basic functions of SQL*Plus
- Use SAP tools for administration of the Oracle database: BR*Tools
- Explain the usage of the SAP CCMS for the administration of Oracle databases

Lesson: Administration of Oracle Instances

Lesson Overview

In this lesson you will learn how to start and stop an Oracle instance, change Oracle parameters, and learn where diagnosis files are stored.



Lesson Objectives

After completing this lesson, you will be able to:

- Change parameters for initialization
- Start and stop the Oracle instance
- Identify and monitor diagnosis files

Business Example

Your SAP EarlyWatch report recommends that you change some Oracle parameters. Because your next maintenance window of the SAP system, which allows a restart of the database, is in two weeks, you want to change the parameters dynamically if possible.

Starting and Stopping the Database

When an Oracle database is started, it goes through three phases:

NOMOUNT

In the NOMOUNT phase, the parameter file is opened and evaluated, and the database instance is started. Operating system resources are allocated using configuration information stored in the parameter file.

MOUNT

In the MOUNT phase, the control files of the database are opened (using the value of parameter `CONTROL_FILES` from the parameter file) and evaluated. The system reads the information about the physical structure of the database. Although data files and log files are not yet opened, parts of the data dictionary are loaded, so V\$ views are available. If one or more of the files specified by the `CONTROL_FILES` initialization parameter does not exist or cannot be opened when you attempt to mount a database, Oracle returns an error message and does not mount the database.

OPEN

In the OPEN phase, all remaining files of the database system are opened. If required, an instance recovery is performed during opening the database. Only when the database is open can valid database users connect to the database. If one or more of the data files or redo log files is not available or cannot be opened when attempting to open a database, Oracle returns an error message and does not open the database.

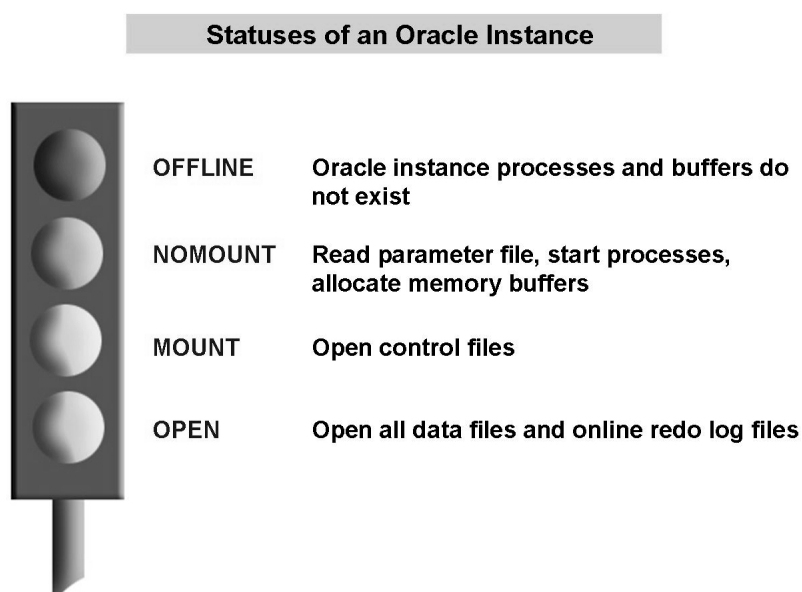


Figure 35: Starting an Oracle Instance

To start the database, start BRTOOLS or BRGUI and choose *Instance management* → *Start up database*. To start the database in batch, use `brspace -c force -f dbstart -s <state>`.



Hint: If the database is in any other state than shutdown when starting the database with BRSPACE, the startup will fail unless you specify the option `-f` | `-force` for the `dbstart` function. In this case, the database will be shut down before starting it up into the requested state.

The instance states NOMOUNT and MOUNT are needed for some special administrative tasks:

- The NOMOUNT state is necessary for creating a database and for re-creating lost control files.
- The MOUNT state is needed for database recovery, for changing the ARCHIVELOG mode, for renaming (moving) data files, and for adding, dropping, or renaming online redo log files.



Hint: BRCONNECT also offers menus and options to start up and shut down the database. The use of this functionality of BRCONNECT is not recommended and should not be used; use BRSPACE instead because it will write log files of its actions.

To stop the database, start BRTOOLS or BRGUI and choose *Instance management* → *Shut down database*. To stop the database in batch, use `brspace -c force -f dbshut -m <mode>`.

The following shutdown modes are available:



NORMAL

No new connections are allowed. Oracle waits for all currently connected users to disconnect from the database. Only after the last user has disconnected (in SAP system: all work processes have been stopped), Oracle shuts down the database: all files are closed, the database is dismounted, and the instance is shut down.

TRANSACTIONAL

After this instruction has been defined, no new connections or new transactions are allowed. Oracle waits for all open transactions to finish, then it disconnects all database connections and shuts down the database.

IMMEDIATE

No new connections are allowed, and no new transactions are allowed to be started after the statement is issued. The PMON process ends all user sessions and performs a rollback of any open transactions. Then the database is shut down.

ABORT

No new connections are allowed, and no new transactions are allowed to be started after the statement is issued. All client SQL statements currently being processed are terminated, without rolling back open transactions. User connections are disconnected and Oracle processes are stopped.

With any of the first three methods, the database is shut down in a consistent state, and no instance recovery is required at the next restart. After SHUTDOWN ABORT, the database data may be inconsistent because of aborted transactions, and the database will require instance recovery at the next startup (which will be performed automatically). Therefore, this method should only be used in exceptional cases, for example, when an Oracle background process terminated abnormally.

Note that NORMAL is the default mode for the Oracle command SHUTDOWN, whereas the BRSPACE default is IMMEDIATE.

While the Oracle commands SHUTDOWN IMMEDIATE and SHUTDOWN ABORT stop the Oracle instance even if work processes in the SAP system have connections to the database, BRSPACE checks for SAP<SCHEMA-ID> or SAPR3 connections and does not continue if any such connections exist. To force the shutdown of the database while SAP is running, use the option `-f` | `-force` to force BRSPACE to shutdown the database.

Initialization Parameters

The instance for an Oracle database is started using an initialization parameter file. Parameters contained in this file configure the system global area and Oracle background processes.

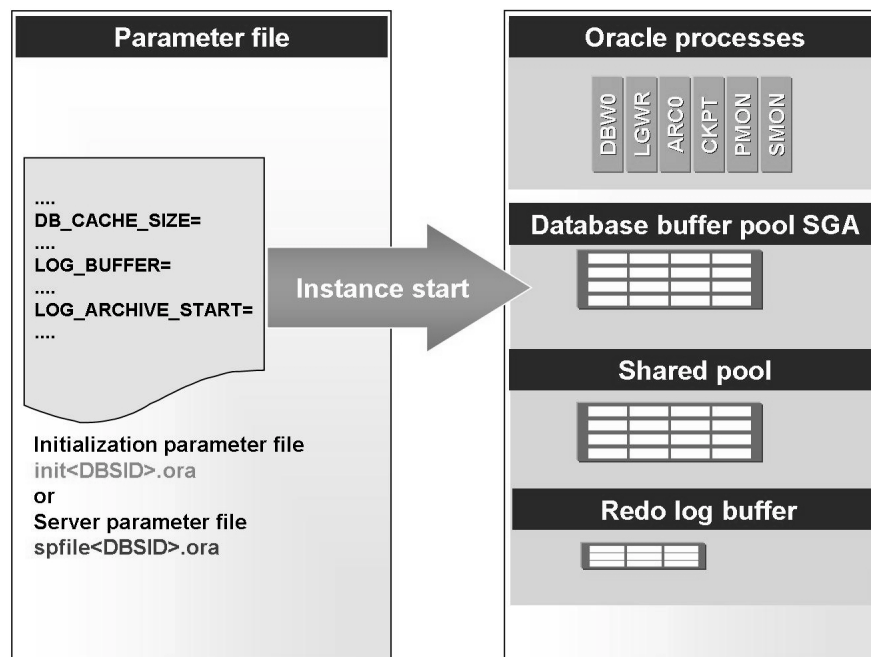


Figure 36: Oracle Instance Profile

Examples of instance parameters:

- `DB_CACHE_SIZE` specifies the size of the buffer pool.
- `LOG_BUFFER` specifies the amount of memory that Oracle uses when buffering redo entries.
- `LOG_ARCHIVE_START=TRUE` specifies that the archive is started automatically.
- `CONTROL_FILES` specifies one or more names of control files.

Oracle has traditionally stored initialization parameters in a text initialization parameter file, usually called `init<DBSID>.ora`. Starting with Oracle 9i, you can choose to maintain initialization parameters in a binary server parameter file (logically called *SPFILE*, the physical name being `spfile<DBSID>.ora` or `spfile.ora`). This is a server-side initialization parameter file, maintained on the machine where the Oracle database server is running.

Oracle allows you to set, change, or delete (restore to default) values of some initialization parameters dynamically. However, if you use the traditional initialization parameter file (`init<DBSID>.ora`), such a change affects only the currently running instance, since there is no mechanism for automatically updating initialization parameters on disk. They must be updated manually and then the instance must be restarted. Using a server parameter file overcomes this limitation because initialization parameters stored in a server parameter file are persistent, in that any changes made to the parameters while an instance is running can persist across instance shutdown and startup.

SPFILE is fully supported by BR*Tools (including parameter maintenance) as of release 6.40. SAP recommends switching to using the *SPFILE* as part of the database upgrade to Oracle 9i.



Caution: If you use *SPFILE*, do not make parameter changes on Oracle level because this only changes parameter values in *SPFILE* itself. With a parameter change in the *SPFILE* using BR*Tools, a new `init<DBSID>.ora` is generated automatically from the *SPFILE* so that the contents of *SPFILE* and `init<DBSID>.ora` remain consistent. Certain SAP transactions (such as DB02 or ST04) still rely on the `init<DBSID>.ora`. If the *SPFILE* and `init<DBSID>.ora` are not kept consistent, these transactions show a status that is out-of-date in comparison with the *SPFILE*. If no `init<DBSID>.ora` exists, no display occurs at all. Because of this mechanism, it is also not possible to keep an old version of `init<DBSID>.ora` in the default location. If you need this, create a copy of `init<DBSID>.ora` on operating system level.

The SAP installation tools create the *SPFILE* as of release 6.40. It is also possible to create the server parameter file in SQL*Plus.

It is recommended to create and store the *SPFILE* in the default location on the database server, which is the same as the default directory for `init<DBSID>.ora` (`$ORACLE_HOME/dbs` on UNIX, and `%ORACLE_HOME%\database` on Windows). This will ease administration of your database. For example, when the Oracle instance starts, it assumes this default location to read the parameter file.



Hint: When an Oracle instance is started without the specification of a special profile (parameter file), it searches the default location for a file with one of the following names (in this order):

1. `spfile<DBSID>.ora`
2. `spfile.ora`
3. `init<DBSID>.ora`

The search is finished as soon as one of the files is found, and the instance is started using that profile, which is normally an *SPFILE*. If no *SPFILE* exists in the default directory, the instance is started with the standard initialization file. This means you can use the init file as a kind of backup for your *SPFILE* – one more reason to keep the init file consistent with the *SPFILE*.

You can create an *SPFILE* from the standard parameter file `init<DBSID>.ora` with the `CREATE SPFILE` command, regardless of the instance state (the database does not have to be started to issue this statement):

```
SQL>CONNECT / AS SYSDBA
SQL>CREATE SPFILE FROM PFILE;
```

With this command, an *SPFILE* is created at the default location (platform-specific) under the default name. Then you must restart the instance so that the newly created server parameter file is used.

If a server parameter file of the same name already exists on the server, it is overwritten with the new information. If the instance is running and it has already been started using a server parameter file, an error is raised if you attempt to recreate the same server parameter file that is currently being used.

You cannot create an *SPFILE* from the current system status of a started instance; it is always created from an existing Oracle parameter file.

You can have several *SPFILEs* on one computer, but only one of these may be active at any given time. When an instance is to be started, the database administrator can use the *SPFILE* parameter to specify a specific server parameter file (possibly with a name different from the default) to be used.



Caution: Starting the database using a different *SPFILE* than the default is not recommended for standard operation. Only use this in emergency situations.

To view **current parameters set in the profile**, start BRTOOLS or BRGUI and choose *Additional functions* → *Show profiles and logs* → *Oracle profile*. To view **all Oracle parameters**, start BRTOOLS or BRGUI and choose *Instance management* → *Show database parameters*. The column *Modif.* shows whether the parameter can be changed dynamically or not.



Caution: Although you can open the binary server parameter file with a text editor and view its text, do not manually edit it. Doing so will corrupt the file. You will not be able to start your instance, and if the instance is running, it could fail.



Note: For more details on server parameter file, see SAP Note 601157.

Maintenance of Oracle Parameters



Parameter classes

Dynamic	Parameter can be modified while the instance is running.
Static	Parameter modifications become effective after a restart of the instance.

Scope of parameter changes

SPFILE	The change is only applied in the server parameter file, so it is effective at next startup, and it is persistent. <u>This is the only scope allowed for static parameters.</u>
MEMORY	The change is only applied in memory, not in the profile (it is not persistent).
BOTH	The change is applied both in memory and in the Server Parameter File (the effect is immediate and persistent).

BRSPACE supports changes of dynamic parameters in the memory, if the Server Parameter File is not used.

Figure 37: Maintenance of Oracle Parameters

Those Oracle parameters that can be immediately modified (while running) in memory are called dynamic. Changes of static parameters need a restart of the Oracle instance.



Hint: Use the “Reference” manual from the Oracle documentation to find details about individual parameters, including the parameter class.

The classification of Oracle parameters into static and dynamic is independent of whether a server parameter file is used or not. As already mentioned, use of a server parameter file just simplifies maintenance of dynamic parameters.

If no server parameter file is used in an instance, a persistent change to a parameter value is necessary by editing the standard initialization file. In this case, BRSPACE only supports modifications of dynamic parameters in memory (like Oracle statement *ALTER SYSTEM* does, which is called by BRSPACE).

To modify an Oracle parameter, start BRTOOLS or BRGUI and choose *Instance management* → *Alter database parameters*. If you already know the parameter to change, enter it in *Database parameter (parameter)* and continue. If you want to select the parameter from a list, do not make any changes. Choose *c* to continue, select *Alter database parameter*, and enter the position number of the parameter (not the name) you want to change from the list of parameters. The following menu is shown:



Changing Database Parameters

BR0657I Input menu 212 - please check/enter input values

Options for alter of database parameter 'log_buffer'

```

1 * Parameter description (desc) ..... [redo circular buffer size]
2 * Parameter type (type) ..... [integer]
3 * Current parameter value (parval) . [2097152]
4 * Value in spfile (spfval) ..... [100000]
5 ? New parameter value (value) ..... []
6 - Scope for new value (scope) ..... [spfile]
7 # Database instance (instance) ..... []
8 ~ Comment on update (comment) ..... []
9 - SQL command (command) ..... [alter system set log_buffer = scope =
spfile]
```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

Information for the parameters:

**desc**

Parameter description from view *V\$PARAMETER*

parval

Current parameter value. Can differ from the value in *SPFILE* when either the parameter was previously changed with scope *MEMORY* (only for dynamic parameters), or Oracle rounded the parameter value on startup.

spfv

Parameter value currently in *SPFILE*

value

New parameter to be set

comment

The comment entered here will be stored in the parameter file.

scope

Can be *SPFILE*, *MEMORY*, or *BOTH*. For static parameters, only *SPFILE* is possible and the parameter will only be activated on restart of the Oracle instance.

After entering all required information and continuing, BRSPACE will execute the corresponding SQL command to change the profile parameter. In addition, it will copy *SPFILE* to the standard init file and will create an entry in *param<DBSID>.log* in the *sapreorg* directory to maintain a history of parameter changes.

Parameter Recommendations

Only optimum database parameter settings can ensure the database runs without errors and good system performance. The parameter recommendations depend on the Oracle release and SAP application you are using.

For Oracle 10g, see SAP Note 830576. This note contains SAP recommendations for the best configuration of the Oracle 10g database in SAP environments. These parameter recommendations are subject to change. We therefore recommend that you check the most recent version of this SAP Note at least once a month, and make any changes that may be necessary.

Previously, some parameter settings for the Oracle database (for example, for the Cost Based Optimizer) depended on whether you were using a normal R/3 system or a BW-based system. As of Oracle 10g, there are standard parameter settings for all systems as described in this SAP Note. Any exceptions are mentioned explicitly.

For Oracle databases 8 and 9i, see SAP Note 124361. This contains parameter recommendations for ERP systems with SAP 4.0B or higher that do not use BW functions. This SAP Note also refers to other notes with parameter recommendations for other SAP applications (BW, CRM, SEM, and so on).

Oracle Error and Diagnosis Files

Informational messages, warnings, and errors are logged by Oracle in **dump files**. These text files can be used for debugging and troubleshooting. Their location is specified in Oracle parameters `BACKGROUND_DUMP_DEST` (default: `$SAPDATA_HOME/saptrace/usertrace`).

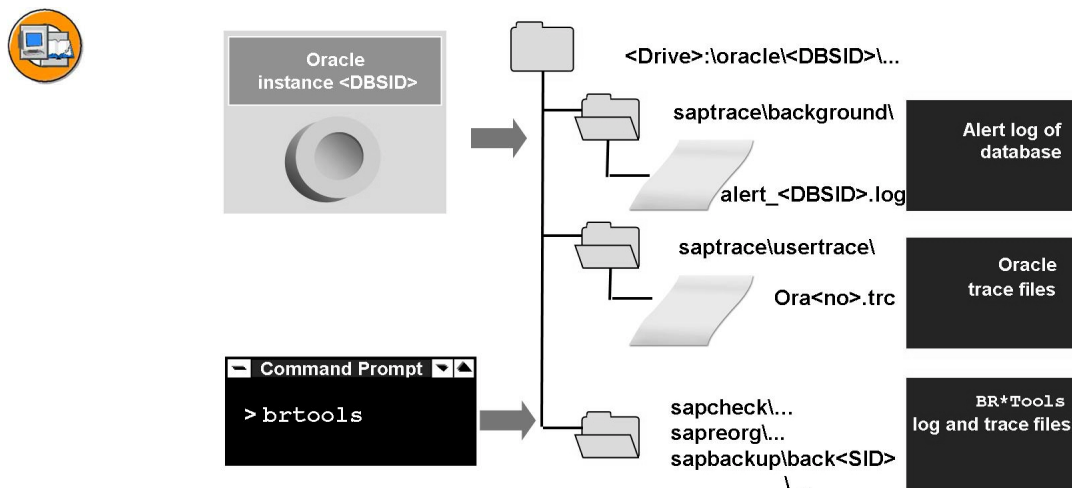


Figure 38: Oracle Error and Diagnosis Files

The `BACKGROUND_DUMP_DEST` directory stores:

- The alert log file `alert_<DBSID>.log`, which logs significant database events and messages, such as startup and shutdown, non-default values of parameters, errors and warnings, log switches and checkpoints, media recovery, creation of tablespaces, and so on.
- Trace files of instance background processes, written especially in case of errors



Hint: The alert file grows without limit, so you may want to delete it periodically. You can delete the file even when the database instance is running.

The `USER_DUMP_DEST` directory contains trace files written on behalf of shadow processes, whenever a problem in a client connection occurs.

Exercise 4: Change Oracle Parameters

Exercise Objectives

After completing this exercise, you will be able to:

- Check and change Oracle parameters

Business Example

You got a recommendation from SAP EarlyWatch to change an Oracle parameter.

Task:

First set BR*Tools to turn archiving on for your database. Then a server parameter file for Oracle will be created from `init<DBSID>.ora` and a parameter is changed.

1. Use BR*Tools to check if archiving is turned on.
2. Use BR*Tools to turn archiving on for your database.
3. Create the server parameter file from `init<DBSID>.ora`.



Caution: After installation of SAP Web AS prior to 6.40, you must create the server parameter file with SQL*Plus. The installation routine for SAP Web AS 6.40 creates the server parameter file during installation automatically.

4. Change parameter `open_cursors` to value **600**. Is it possible to perform this dynamically? Check the setting before and after restarting the database.



Caution: This parameter is set to a value that is not recommended for SAP systems. The database system check will show a warning about the wrong value.

Solution 4: Change Oracle Parameters

Task:

First set BR*Tools to turn archiving on for your database. Then a server parameter file for Oracle will be created from `init<DBSID>.ora` and a parameter is changed.

1. Use BR*Tools to check if archiving is turned on.
 - a) Call BRGUI or BRTOOLS and choose *Instance management* → *Instance status* (confirm twice with “continue”).
 Information about the status of database instance T99


```

1 - Instance number (number) ..... 1
2 - Instance thread (thread) ..... 1
3 - Instance status (status) ..... OPEN
4 - Instance start time (start) ..... 2007-11-26 12.14.00
5 - Oracle version (version) ..... 10.2.0.2.0
6 - Database creation time (create) .... 2007-11-26 09.21.31
7 - Last resetlogs time (resetlogs) .... 2007-11-26 09.21.31
8 - Archivelog mode (archmode) ..... NOARCHIVELOG
9 - Archiver status (archiver) ..... STOPPED
10 - Current redolog sequence (redoseq) . 54
11 - Current redolog SCN (redoscn) ..... 212369
12 - Number of SAP connections (sapcon) . 0
          
```
 - b) The output shows that the archiver is started, but the archive log mode is NOARCHIVELOG, which means that archiving is turned off.
2. Use BR*Tools to turn archiving on for your database.
 - a) Call BRGUI or BRTOOLS and choose *Instance management* → *Alter database instance* → *Set archivelog mode*. Your database instance will be stopped and restarted into mount state, archive log mode will be turned on, and the database will be opened.
3. Create the server parameter file from `init<DBSID>.ora`.

Continued on next page



Caution: After installation of SAP Web AS prior to 6.40, you must create the server parameter file with SQL*Plus. The installation routine for SAP Web AS 6.40 creates the server parameter file during installation automatically.

a)

```
G:\oracle\T99>sqlplus / as sysdba
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 12:22:43 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
Connected to:
```

```
Oracle Database 10g Enterprise Edition Release 10.2.0.2.0 - 64bit  
Production
```

```
With the Partitioning, OLAP and Data Mining options
```

```
SQL> create spfile from pfile
```

```
File created.
```

```
SQL> exit
```

- b) Do not forget to restart your database after this step. To do so, use BRGUI or BRTOOLS and choose *Instance management* → *Shut down database* resp. *Instance management* → *Start up database*.
4. Change parameter `open_cursors` to value **600**. Is it possible to perform this dynamically? Check the setting before and after restarting the database.

Continued on next page



Caution: This parameter is set to a value that is not recommended for SAP systems. The database system check will show a warning about the wrong value.

- a) Start BRGUI or BRTOOLS and choose *Instance management* → *Alter database parameters*.
- b) In the next input menu, BRSPACE options for alter database parameter, enter the database parameter `open_cursors` and choose *Continue*.
- c) In the following menu, Options for alter of database parameter 'open_cursors', enter the new value of **600**.
- d) Because scope both is offered, the parameter change will be done in memory (dynamically) and in the server profile:

Options for alter of database parameter 'open_cursors'

```

1 * Parameter description (desc) ..... [max # cursors per session]
2 * Parameter type (type) ..... [integer]
3 * Current parameter value (parval) . [800]
4 * Value in spfile (spfval) ..... [<same>]
5 - New parameter value (value) ..... [600]
6 - Scope for new value (scope) ..... [both]
7 # Database instance (instance) ..... []
8 ~ Comment on update (comment) ..... []
9 - SQL command (command) ..... [alter system set
    open_cursors = 600 scope = both]

```

- e) To check if the parameter was really changed dynamically as well as permanently, check the parameter setting before and after restart of the database instance. Use BRGUI or BRTOOLS and choose *Instance management* → *Show database parameters*.



Lesson Summary

You should now be able to:

- Change parameters for initialization
- Start and stop the Oracle instance
- Identify and monitor diagnosis files



Unit Summary

You should now be able to:

- Describe the architecture and the main components of an Oracle database
- Explain the basic concepts of the Oracle database
- Identify the file structure of an Oracle database in a SAP system
- Name the default operating system and database users
- Explain Oracle communication over a network (NETServices)
- Describe the function of the Oracle listener
- Start and stop the Oracle listener
- Identify the main tools for administration of the Oracle database
- Describe basic functions of SQL*Plus
- Use SAP tools for administration of the Oracle database: BR*Tools
- Explain the usage of the SAP CCMS for the administration of Oracle databases
- Change parameters for initialization
- Start and stop the Oracle instance
- Identify and monitor diagnosis files



Test Your Knowledge

1. The system global area consists of the _____, the _____, and the _____.

Fill in the blanks to complete the sentence.

2. Which of the following statements is true?

Choose the correct answer(s).

- ☐ A Data files can be mirrored by Oracle.
- ☐ B Online redo log files can be mirrored by Oracle.
- ☐ C Control files can be mirrored by Oracle.
- ☐ D Parameter files can be mirrored by Oracle.

3. For minimal security of an Oracle database:

Choose the correct answer(s).

- ☐ A The database must be installed on a cluster.
- ☐ B The online redo log files must be mirrored.
- ☐ C Online redo log files and data files must reside on different disks.
- ☐ D Archiving must be turned on.
- ☐ E Offline redo log files and data files must reside on different disks.

4. Which files are stored in directory %ORACLE_HOME%\database (on Windows), and \$ORACLE_HOME/dbs (on UNIX)?

Choose the correct answer(s).

- ☐ A Control files
- ☐ B Oracle profiles
- ☐ C BR*Tools profiles
- ☐ D Oracle trace and log files
- ☐ E BR*Tools log files

5. Which users are standard Oracle users created by the Oracle installer?

Choose the correct answer(s).

- ☐ A SYSTEM
- ☐ B SYS
- ☐ C SAP<SCHEMA-ID>
- ☐ D OPSS<HOSTNAME>/<SAPSID>ADM (Windows),
OPSS<SAPSID>ADM (UNIX)
- ☐ E SYSDBA

6. Which users are standard Oracle users created by the SAP installation tool?

Choose the correct answer(s).

- ☐ A SYSTEM
- ☐ B SYS
- ☐ C SAP<SCHEMA-ID>
- ☐ D OPSS<HOSTNAME>/<SAPSID>ADM (Windows),
OPSS<SAPSID>ADM (UNIX)
- ☐ E SYSDBA

7. The correct procedure to change the password of database user SAP<SCHEMA-ID> is to use the Oracle command ALTER USER.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

8. The Oracle tool to test the connection to the listener is called _____. To test the connection to the database with SID C11 on host twdf0505, enter _____ at the operating system level. If you detect that the listener is not running on the database server, you can use the command _____ on Windows and UNIX to start the listener.

Fill in the blanks to complete the sentence.

9. Which of the following tools does not have its own menus?

Choose the correct answer(s).

- ☐ A BRTOOLS
- ☐ B BRBACKUP
- ☐ C BRCONNECT
- ☐ D BRSPACE
- ☐ E BRRECOVER

10. Because the interactive programs of BR*Tools provide their own menus, they cannot be called from BRTOOLS or BRGUI.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

11. If you know the correct SQL command, we recommend that you call this command (for example, as seen in BR*Tools) from SQLPLUS to perform changes to the database.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

12. All recommended regular activities for checks, housekeeping and backups can be scheduled in the DBA Planing Calendar (transaction DB13)

Determine whether this statement is true or false.

- ☐ True
- ☐ False

13. After creating the server profile from init<DBSID>.ora, you can change Oracle profile parameters in both the server profile and init<DBSID>.ora.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

14. When using the server parameter file

Choose the correct answer(s).

- ☐ A After changing parameters with scope BOTH, the database must be restarted.
- ☐ B After changing parameters with scope MEMORYT, the database must be restarted.
- ☐ C After changing parameters with scope SPFILE, the database must be restarted.
- ☐ D With scope MEMORY, the previous parameter value is used after restart of the database instance.

15. When stopping the database using shutdown mode _____, instance processes are stopped immediately and no rollback of open transactions is performed during the shutdown.

Fill in the blanks to complete the sentence.

16. Which of the following shutdown modes leaves the database in a consistent state?

Choose the correct answer(s).

- ☐ A NORMAL
- ☐ B TRANSACTIONAL
- ☐ C IMMEDIATE
- ☐ D ABORT

17. The Oracle alert log file is located in directory _____.

Fill in the blanks to complete the sentence.



Answers

1. The system global area consists of the database buffer, the redo log buffer, and the shared pool.

Answer: database buffer, redo log buffer, shared pool

2. Which of the following statements is true?

Answer: B, C

3. For minimal security of an Oracle database:

Answer: B, D, E

4. Which files are stored in directory %ORACLE_HOME%\database (on Windows), and \$ORACLE_HOME/dbs (on UNIX)?

Answer: B, C

5. Which users are standard Oracle users created by the Oracle installer?

Answer: A, B

6. Which users are standard Oracle users created by the SAP installation tool?

Answer: C, D

7. The correct procedure to change the password of database user SAP<SCHEMA-ID> is to use the Oracle command ALTER USER.

Answer: False

8. The Oracle tool to test the connection to the listener is called TNSPING. To test the connection to the database with SID C11 on host twdf0505, enter tnsping C11 at the operating system level. If you detect that the listener is not running on the database server, you can use the command lsnrctl start on Windows and UNIX to start the listener.

Answer: TNSPING, tnsping C11, lsnrctl start

9. Which of the following tools does not have its own menus?

Answer: B, C

These tools must be either called from the command line with all necessary options and parameters, or started from BRTOOLS.

10. Because the interactive programs of BR*Tools provide their own menus, they cannot be called from BRTOOLS or BRGUI.

Answer: False

All functional programs can be called from BRTOOLS or BRGUI.

11. If you know the correct SQL command, we recommend that you call this command (for example, as seen in BR*Tools) from SQLPLUS to perform changes to the database.

Answer: False

Using SQLPLUS is only necessary in rare cases, for example to execute scripts provided by SAP. Executing SQL commands seen in BR*Tools is dangerous, because BR*Tools usually perform additional actions and checks before and after executing the SQL command.

12. All recommended regular activities for checks, housekeeping and backups can be scheduled in the DBA Planing Calendar (transaction DB13)

Answer: True

13. After creating the server profile from init<DBSID>.ora, you can change Oracle profile parameters in both the server profile and init<DBSID>.ora.

Answer: False

Changes are only allowed in the server profile, preferably through BR*Tools, as init<DBSID>.ora is regularly recreated from the server profile.

14. When using the server parameter file

Answer: C, D

15. When stopping the database using shutdown mode abort, instance processes are stopped immediately and no rollback of open transactions is performed during the shutdown.

Answer: abort

16. Which of the following shutdown modes leaves the database in a consistent state?

Answer: A, B, C

17. The Oracle alert log file is located in directory \$SAPDATA_HOME/sap-trace/background.

Answer: \$SAPDATA_HOME/saptrace/background

Unit 2

Backup, Restore & Recovery

Unit Overview

In this unit, you will learn how to plan and perform backups, as well as performing a restore and recovery of the database.



Unit Objectives

After completing this unit, you will be able to:

- Explain the importance of backups
- List the different backup types (offline, online, partial, and incremental backup)
- Explain the special importance of backups of the archived redo log files
- Define a backup strategy depending on database size, tape capacity, and available time for restore/recovery
- Identify the different SAP tools for backup, restore, and recovery
- Explain the concept of Oracle's Recovery Manager (RMAN)
- Customize the SAP tools
- Describe tape management with BR*Tools
- Initialize and manage backup tapes with BR*Tools
- Perform online, offline, and partial backups
- Perform RMAN backups, including incremental backups
- Create backups of archived redo log files
- Describe a potential problem that would lead to a restore/recovery scenario
- Perform a complete recovery of the database
- Perform a point-in-time recovery of the database
- Perform a disaster restore/recovery of the database
- Explain the various backup strategies supported by SAP
- Decide which strategy fits your needs

Unit Contents

Lesson: Backup Strategy	109
Exercise 5: Backup Strategy	123
Lesson: Backup Tools	126
Exercise 6: Backup Tools	147
Lesson: Appendix: Tape Management with BR*Tools	153
Lesson: Performing Backups	171
Exercise 7: Performing Backups	189
Lesson: Restore and Recovery	192
Exercise 8: Restore and Recovery	213
Lesson: Advanced Backup Techniques	220

Lesson: Backup Strategy

Lesson Overview

Before performing backups, the backup strategy has to be planned and tested. In this lesson, you will learn to define a backup strategy for different database sizes.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain the importance of backups
- List the different backup types (offline, online, partial, and incremental backup)
- Explain the special importance of backups of the archived redo log files
- Define a backup strategy depending on database size, tape capacity, and available time for restore/recovery

Business Example

Until now, you performed a daily offline backup of the database starting at 8pm, but now that employees of your subsidiaries in Singapore, Boston, and Poland also access the SAP system, it must be constantly online. You need a new backup strategy that enables the SAP system to be up 24 hours per day.

Importance of Database Backups

With databases, performing a proper backup requires special actions. Compared to normal backup of files, a database is a collection of files that are dependant on each other. If you lose a single file of the database, simply restoring it is rarely sufficient.

Reasons for Data Loss

Business application data of an SAP system, stored in a relational database, is usually very dynamic and requires a comprehensive security strategy. If you do not have a suitable backup strategy, external factors, physical errors, and logical errors can cause system downtime and may lead to data loss.

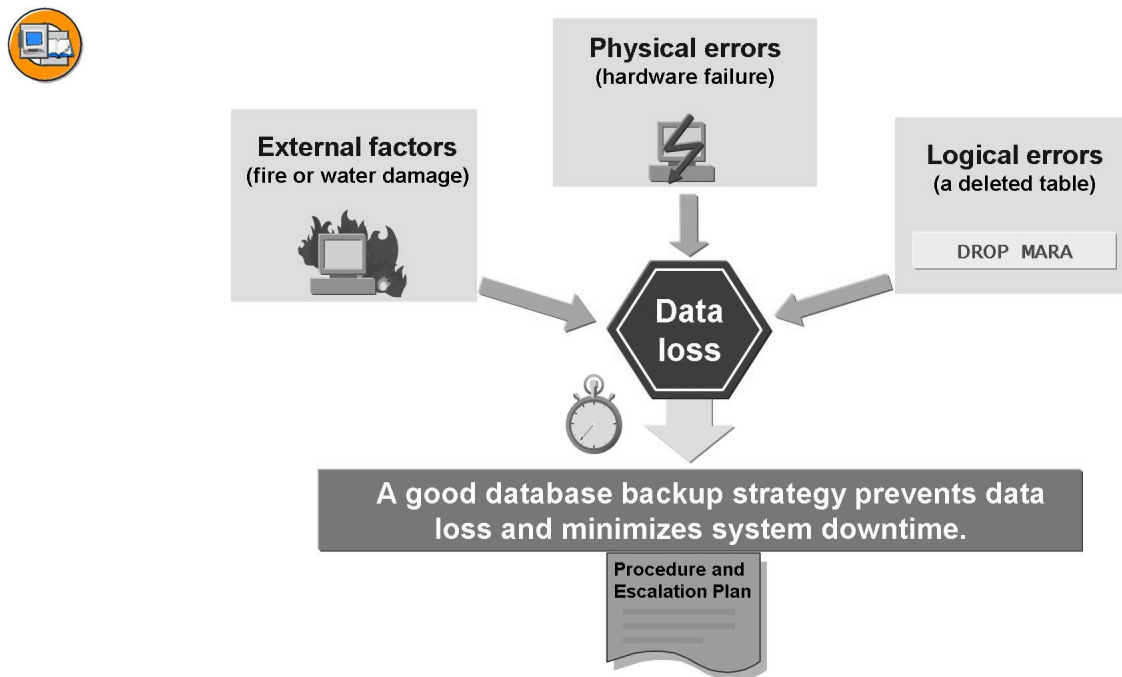


Figure 39: Importance of Backups

If data is lost due to external factors, such as water damage to your hardware, or physical errors, such as hardware failure, you must recover the database up to the point in time when the database crashed. If this complete recovery is possible, only the data of transactions uncommitted at the time of error will be lost.

If data is lost due to logical errors, such as unintentionally deleting a table, you may have to recover the database up to a point in time shortly before the error occurred.

Your backup strategy must be designed according to your company's needs. To ensure the availability of your SAP system, your backup strategy must be carefully tested before your SAP system goes live, and again after any changes to your backup strategy.

When planning your backup strategy, take into account the maximum downtime for each of the above recovery scenarios.

To ensure that the correct steps are performed for each of the scenarios, create a document containing organizational descriptions of procedures and an escalation plan. This document must be understood by the person who performs database restore and recovery.

You should evaluate and implement the most suitable backup type and method for your company. SAP provides tools that support different types of backups, such as online backups, incremental backups with RMAN, or split-mirror backups.

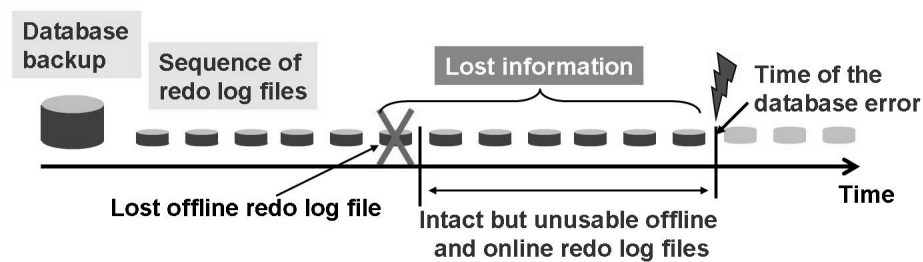
Importance of Redo Log Files

When you want to perform a complete recovery, to recover the data to the (committed) state that the data had at the moment of crash, you need all the offline and online redo log files written from the point in time of the last database backup.



Forward recovery

A database backup is restored and
you want to recover data from redo log files



**If one offline redo log file is lost,
none of the files that follow it can be used.**

Figure 40: Importance of Redo Log Files

If a file is missing from the chain of offline redo log files, a restore of subsequent offline redo log files and corresponding recovery of the database is not possible. You can then only perform a point-in-time recovery, using all offline redo log files older than the lost one. This will result in a loss of data changes performed from the point in time of your lost offline redo log file. Therefore, you should keep at least two copies of all offline redo log files on disks or on tapes.

Disaster Recovery

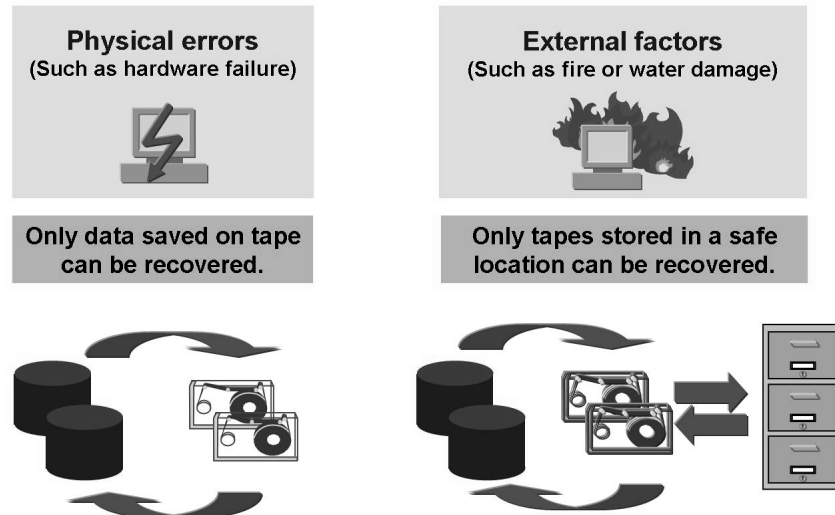


Figure 41: Disaster Recovery

If you have a hardware failure, you can lose one disk, a disk subsystem, or the complete hardware. In this case, only data backed up on external media, such as tapes, can be restored. This includes the offline redo log files; redo log information that is not stored on tapes may be lost.

If data loss occurs due to external factors, such as fire or water damage, all backup media that is not stored in a safe location may be lost.

Recovery from a Logical Error

Logical errors can be caused by:

- Manually dropping database objects
- Manually deleting parts of a database objects, such as rows in a table
- Application errors



Point-in-time recovery

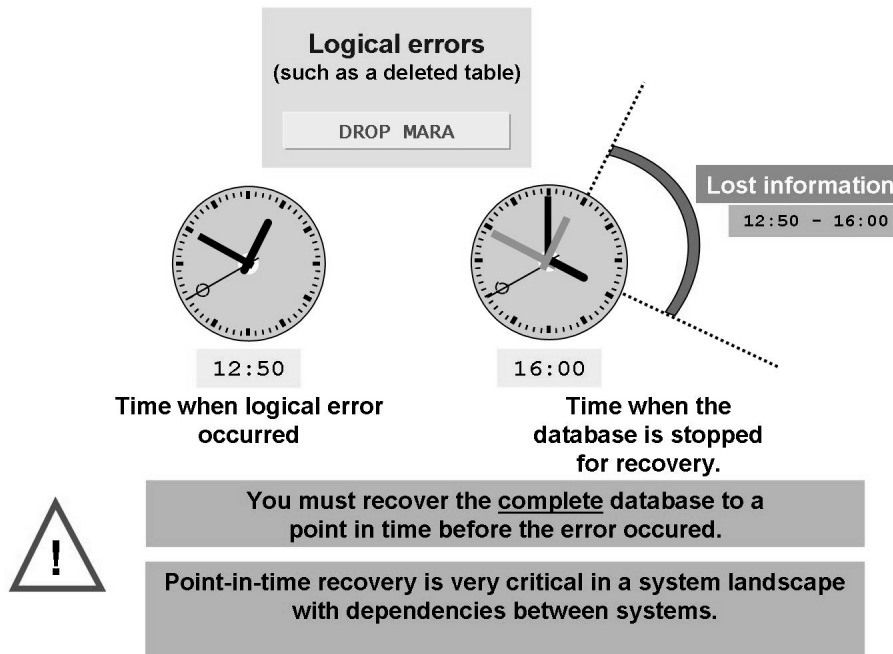


Figure 42: Recovery from a Logical Error

Point-in-time recovery may be needed in other situations, too, for example when you need to reset your system to a state before an upgrade, and a complete database backup corresponding to that point in time is not available.

When performing point-in-time recovery, you must recover the complete database, since the data from different tables must be consistent. Consequently, all data changes made between the time chosen for point-in-time recovery and the time the database is stopped for recovery is lost. This is especially critical in a system landscape with dependent data stored in two or more systems: point-in-time recovery performed in one of the systems results in data inconsistencies.

For these reasons, point-in-time recovery should not be thought of as a standard solution for logical errors in production systems. Depending on the table, it may be possible to restore and recover the database on a different system (on a different computer), and then import the missing table or the missing table rows from that system to your production system. This method avoids data loss, but it requires expert knowledge of the application module that uses the table.

Verification of Data

Your backup strategy should include verifying the data to be backed up as well as the data on tapes.



Verify regularly both data to be backed up and data on backup media!

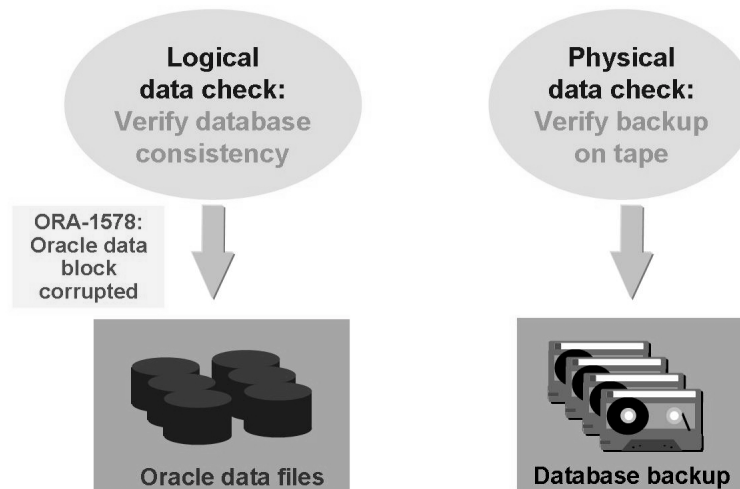


Figure 43: Verification of Data

To verify the consistency of the Oracle database, perform a **logical data check**, which discovers corrupt data blocks:

- Corrupt Oracle blocks (error ORA-1578) can appear in your database as a result of operating system or hardware errors.
- Without the logical data check, corrupt data blocks are only detected when Oracle processes access these data blocks while attempting to access a table within the database. Corrupt blocks that are accessed rarely may remain undetected in your system for a long time.
- Corrupt Oracle blocks are not recognized during a backup; therefore, a database backup can contain corrupt blocks. They make the backup unusable because they are restored in the database in exactly the same state.

Logical data checks should be performed at regular intervals, preferably once a week. For optimal performance, carry out this check during periods of low system activity, for example, on weekends.

To verify tapes used for a database backup, perform a **physical data check**. During this check, the tapes are read and physical correctness of the data transferred is examined.

At the end of an offline backup, you can check at binary level whether the files read from tape are identical to those in the database. This requires that the database remains closed during the procedure.

After an online backup, during which data changes can occur in the database, you can only check that all files on the tapes are readable.

Backup cycle

A backup cycle is a time period during which you keep the backups on your tapes. The length of the backup cycle is the retention period for your tapes; a tape is reused only if the backup on it is older than the retention period.

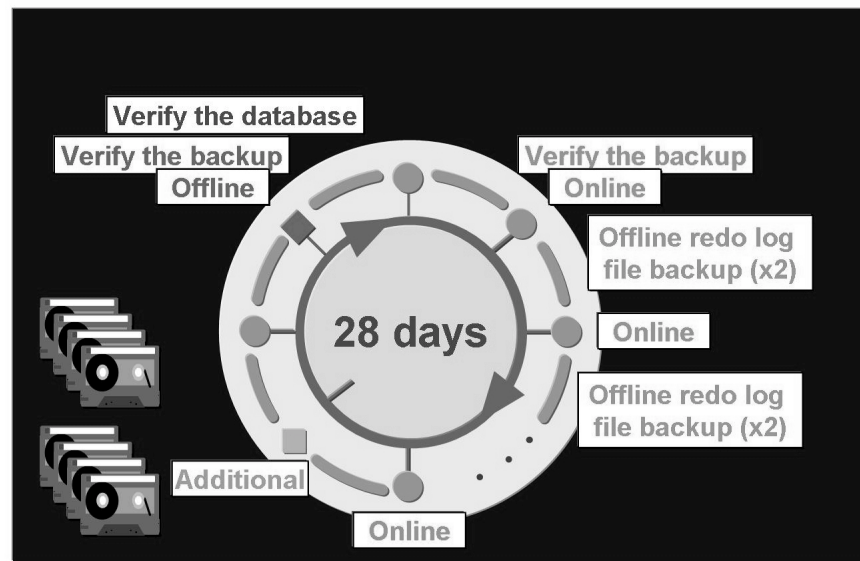


Figure 44: Backup Cycle

SAP recommends a backup cycle of four weeks. The backup cycle must be the same for database backups and offline redo log backups:

- Perform a complete online backup each work day.
- Perform a complete offline backup at least once in the cycle.
- Back up the offline redo log files on each work day, as well as after every online and offline backup. Ensure that you back up every offline redo log file twice, on separate tapes, before the file is deleted in the archive directory.
- To verify a backup, carry out a logical check of the database before or after the backup, and check the backup for physical errors. You must perform backup verification at least once in the backup cycle. However, it is recommended to do it once a week.
- Remove the last verified full offline backup of each cycle from the tape pool, and keep this backup in long-term storage. The removed tapes must be replaced with new ones.

The relation between the backup cycle length and frequency of complete database backups should be such that you always have several generations of complete backups. This protects you from data loss even in a situation where your last database backup cannot be found or is unusable.

Changes to the database file structure affect the subsequent database restore. These changes occur when a data file is added, when a data file is moved to a different location, or when a tablespace and its data files are reorganized. Perform additional backups after each database reorganization, and after each system upgrade. Place these additional backups in long-term storage.

You can also perform additional backups after database structure modification, but this is not necessarily required. BRRECOVER handles such modifications automatically during recovery.

Backup Types

There are two main methods of backing up an Oracle database.

- User-managed backups in which the administrator performs a manual backup on the operating system. In this method, backups with operating system errors are created, and are written back if a recovery is necessary.
- A server-managed backup using the Oracle Recovery Manager. The Oracle Recovery Manager performs the backups using shadow processes on the Oracle server.

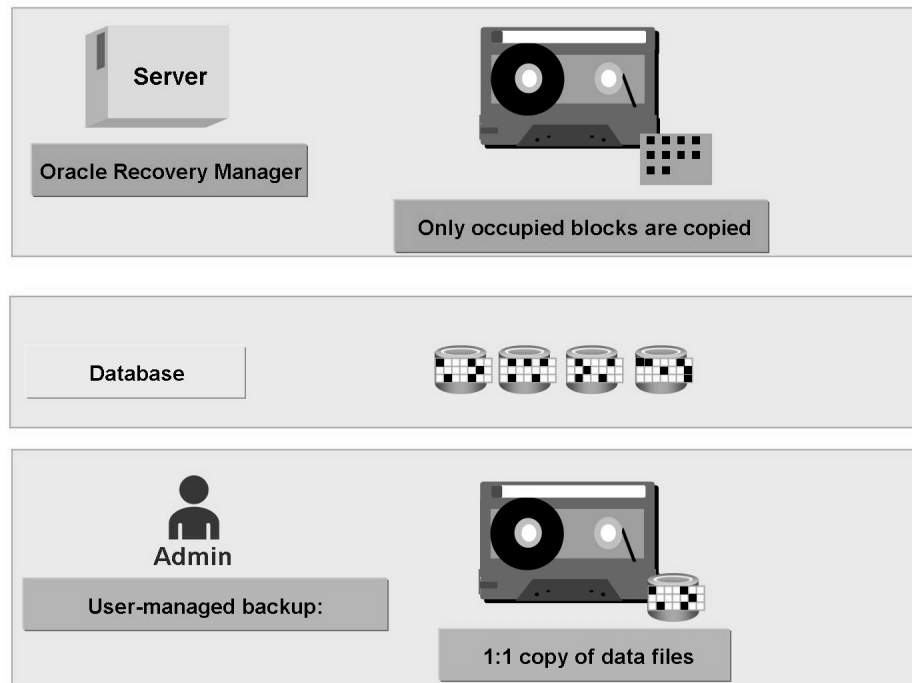


Figure 45: Backup Methods

When using these backup methods, different backup types are available on two levels for an Oracle database. These levels correspond to the following questions:

- Is the database closed or open during the backup procedure?
- Is the database backed up completely and, if not, what part of it is backed up?

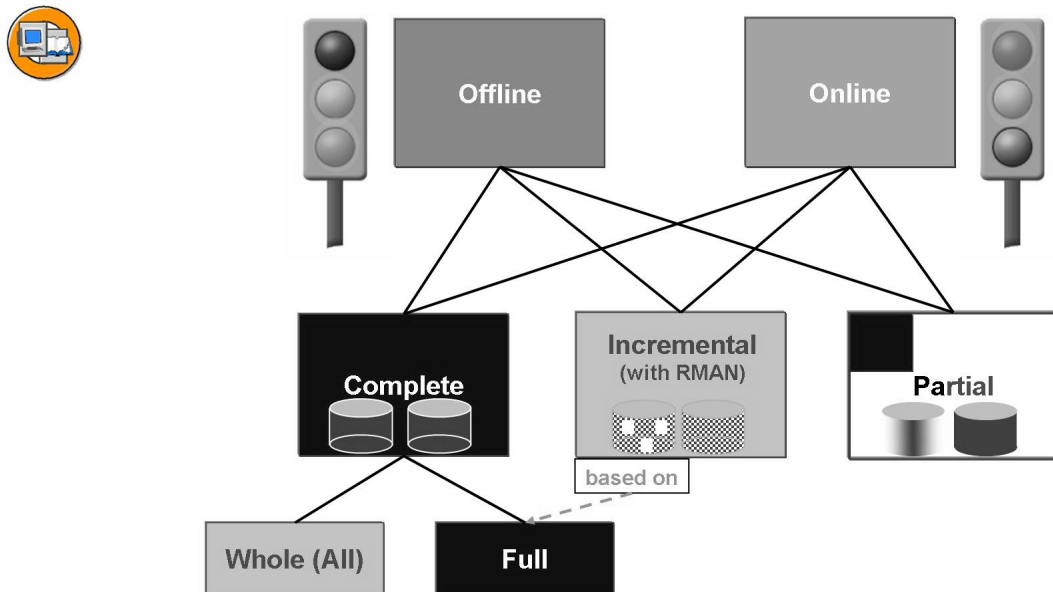


Figure 46: Backup Types

Offline Backup

The database is shut down before the backup. Database files are therefore copied to the backup media in a closed state. If the database has been shut down consistently (standard when using SAP tools), data is backed up in a consistent state, and can be restored and opened even without redo log files.

Online Backup

The database remains open during the backup, and activities can take place (data can be modified) during this time. Obviously, data blocks copied to the backup medium at the beginning of the backup procedure may correspond to an older system change number than those backed up at the end. To restore a consistent database from such a backup, you need at least the redo information written during the time of the backup procedure, so that all blocks in the restored database can be recovered to a state corresponding to the time of the end of the backup.



Hint: Performance of the system may go down slightly during an online backup, so online backups should be scheduled at times of low activity.

Perform a Complete Backup

All data in the database is backed up. If you perform a **full backup**, after backing up all data in the database, an additional piece of information (the catalog information) is written to the control file by the Recovery Manager, which makes it possible to subsequently create incremental backups. A **whole backup** creates a backup of all data without the catalog information. In terms of the database data, there is no difference between the whole backup and full backup.



Hint: When performing a full backup, RMAN is always used to create and append the backup catalog information to the control file. The backup itself can be performed with or without RMAN; this depends on further backup settings (`tape_copy_command` and `backup_dev_type`).

Incremental Backup

If you have created a full backup, you can later back up just those data blocks that have changed since the time of the full backup. This will reduce the amount of data to be backed up; it does not significantly shorten the backup time, because to check if a database block was changed and so must be backed up, the block must be read.

- An incremental backup can only be based on a previous full backup.



Caution: An incremental backup is useless if the corresponding full backup is already overwritten. When performing incremental backups it is therefore highly recommended to perform at least one full backup per week **and** to perform four full backups per backup cycle.

- Like full backup, incremental backup is controlled by RMAN, which uses the control file for this purpose.
- With SAP tools, only cumulative incremental backup is supported. This means that each incremental backup saves all blocks modified since the last full backup, not since the last incremental backup.
- An incremental backup performed with SAP tools is always a backup of the whole database. You cannot choose individual data files to be backed up with an incremental backup.
- When performing an incremental backup, Oracle must read all blocks of all data files to check which ones have been changed. Therefore, an incremental backup can reduce the backup time only if a long backup runtime was caused by low throughput on the tape stations.

Partial Backup

If a complete backup takes too long, you may choose to back up the database in smaller parts. The sum of individual partial backups must, of course, cover the entire database, for example, during one week.



Caution: Although both Oracle and SAP tools support the recovery of data files from different backup runs, this type of recovery requires all offline and online redo log files generated since the backup of the oldest data files, and the recovery process may be very time consuming.

Defining a Backup Strategy

Depending on factors such as database size, tape size, availability, and other factors, you should define a suitable backup strategy.

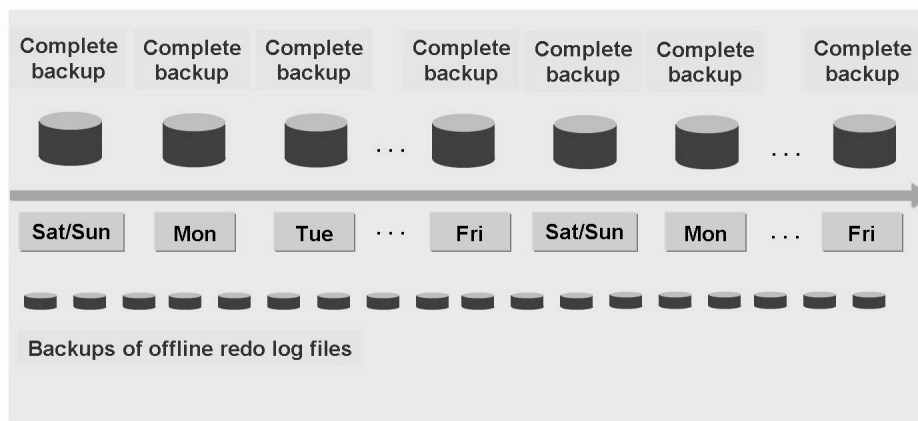


Figure 47: Backup Strategy: Example 1

You can use a simple backup strategy consisting of regular complete database backups and offline redo log file backups. To recover from a media error (disk), you restore missing database files from a complete backup (preferably from the last one), restore offline redo log files written during and after this backup (those that have already been deleted from the archive directory), and completely recover the database.

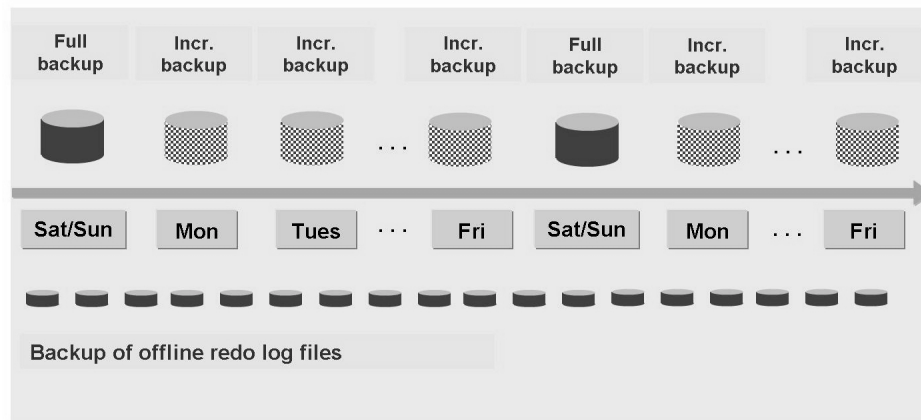


Figure 48: Backup Strategy: Example 2

In a system where incremental backups can reduce backup time, you can perform complete backups less often and replace them with incremental backups. The complete backups must be full backups. To recover from a disk error, you restore missing database files from the last full backup, update them with a restore from the last incremental backup, and recover with help of redo information written during and after the last incremental backup.

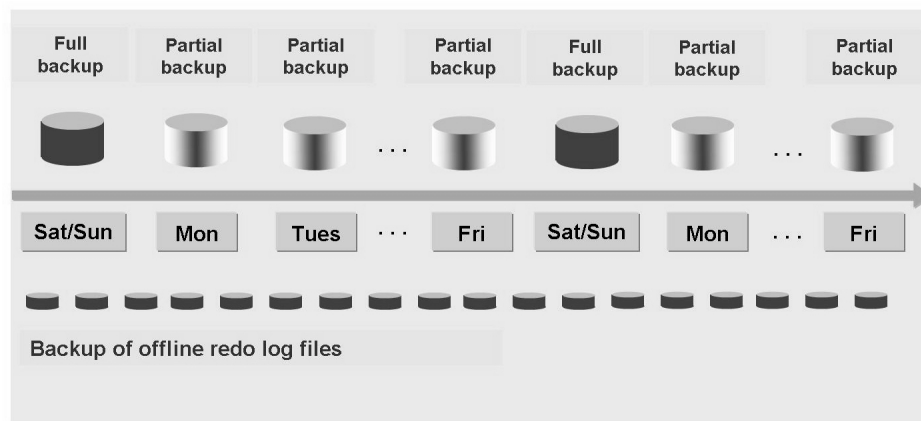


Figure 49: Backup Strategy: Example 3

If you replace a full backup with a partial backup during a week, the runtime of the individual backups is reduced. The recovery may take longer in the event of errors on the storage medium. For example, if a crash occurs on Thursday and the last backup containing lost files is from Monday, you must redo all data modifications performed in these files since Monday.

Exercise 5: Backup Strategy

Exercise Objectives

After completing this exercise, you will be able to:

- Create a proper backup strategy

Business Example

You need to plan a good backup strategy.

Task:

Evaluate the following backup strategy.

1. Technical specifications: The planned size of the database is roughly 100 GB. A maximum of 50 online redo log files of 20 MB are expected to be written daily. Three tape devices are available and each can write or read up to 6 GB per hour. The tapes have a capacity of 40 GB. It takes, on average, three minutes to apply an offline redo log file during the recovery.

Strategy: An online backup is performed every night. Three tapes are reserved for each night. The database administrator performs a backup of the offline redo log files daily, and deletes the offline redo log files from disk afterwards.

Is this a good backup strategy?

Can a full restore be performed in 8.5 hours?

What is the significance for an instance recovery if the error that led to the restore and recovery operation occurred during a long background processing job without a commit?

Solution 5: Backup Strategy

Task:

Evaluate the following backup strategy.

1. Technical specifications: The planned size of the database is roughly 100 GB. A maximum of 50 online redo log files of 20 MB are expected to be written daily. Three tape devices are available and each can write or read up to 6 GB per hour. The tapes have a capacity of 40 GB. It takes, on average, three minutes to apply an offline redo log file during the recovery.

Strategy: An online backup is performed every night. Three tapes are reserved for each night. The database administrator performs a backup of the offline redo log files daily, and deletes the offline redo log files from disk afterwards.

Is this a good backup strategy?

Can a full restore be performed in 8.5 hours?

What is the significance for an instance recovery if the error that led to the restore and recovery operation occurred during a long background processing job without a commit?

- a) The problem when using this backup strategy is that only one copy of the archived redo log files is written to tape before deletion. We recommend that at least two copies are written to different backup media. The data is distributed automatically by BRBACKUP across the tape devices so that the backup can be performed unattended, even if the files are not compressed.

If the data volume is distributed over the three backup media, each tape will contain approximately 33 GB. At a read rate of 6 GB per hour, a restore operation would take approximately 5.5 hours. 1 GB of offline redo log files can be restored in 10 minutes. It takes approximately three minutes to update the redo information to one single redo log file on the database. Therefore, it would take approximately 150 minutes to carry out a recovery with all offline redo log files from one day. To restore and recover the database would take up to 8 hours and 10 minutes. However, this time does not include the time to analyze and repair the error that led to the restore. Additionally, the time of the instance recovery that is performed at system startup is not accounted for in this calculation. Due to the times not accounted for in the calculation, it is unlikely that the full restore and recovery can be performed in 8.5 hours.

An uncommitted transaction has to be rolled back during instance recovery. Therefore, the database needs more time to complete the recovery.



Lesson Summary

You should now be able to:

- Explain the importance of backups
- List the different backup types (offline, online, partial, and incremental backup)
- Explain the special importance of backups of the archived redo log files
- Define a backup strategy depending on database size, tape capacity, and available time for restore/recovery

Lesson: Backup Tools

Lesson Overview

In this lesson, we will introduce the tools BRBACKUP and BRARCHIVE, which are used to back up the database.



Lesson Objectives

After completing this lesson, you will be able to:

- Identify the different SAP tools for backup, restore, and recovery
- Explain the concept of Oracle's Recovery Manager (RMAN)
- Customize the SAP tools

Business Example

After defining your backup strategy and providing the necessary number of tapes, you want to perform the backups. To do this, you must first modify the BR*Tools parameter for backup and restore.

SAP Tools for Backup, Restore and Recovery

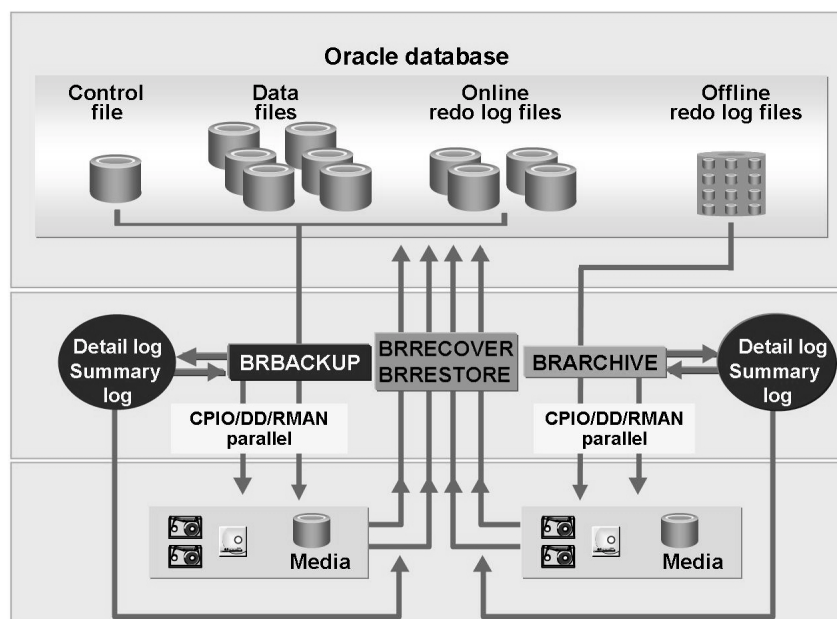


Figure 50: SAP Tools for Backup, Restore and Recovery

The Oracle administration tools delivered by SAP include programs for backing up database files and other files of an SAP system, as well as programs for restoring missing files and recovering data files to a consistent state:

- The program BRBACKUP backs up Oracle data files, the control file, and the online redo log files, where necessary. BRBACKUP can also be used for backing up the Oracle software directories and the SAP system directories.
- The program BRARCHIVE backs up the offline redo log files of the database.
- The program BRRESTORE can restore all files belonging to the database system from the backups: both database files and offline redo log files
- The interactive program BRRECOVER checks the database for missing files, calls BRRESTORE for restoration of missing data files or for restoration of offline redo log files needed for recovery, performs the recovery, and opens the database.

Both BRBACKUP and BRARCHIVE record the performed actions in log files. These log files are analyzed and used by BRRESTORE for restoration of missing files.

BRBACKUP and BRARCHIVE support backups to tape, backups to disk, and backups with third-party tools.

Customizing SAP Backup and Restore Tools

The configuration (initialization) profile `init<DBSID>.sap` contains parameters that determine how BR*Tools (BRBACKUP, BRARCHIVE, BRRESTORE, and so on) perform various functions. It is usually stored in directory `$ORACLE_HOME/dbs` (UNIX) or `%ORACLE_HOME%\database` (Windows).



Profile for SAP tools: init<DBSID>.sap

Parameter examples for the backup and its values:

```

backup_mode           = all | full | incr | <tablespace_name> | ...
backup_type          = offline | offline_force | online | online_cons | ...
backup_dev_type      = tape | disk | util_file | util_file_online | ...
tape_copy_cmd        = cpio | dd | rman | ...
disk_copy_cmd        = copy | dd | rman | ocopy | ...
expir_period         = 28
tape_use_count       = 100
volume_backup        = (<DBSID>B01, <DBSID>B02, ...)
tape_size            = 32G
tape_address         = /dev/rmt/0mn
tape_address_rew     = /dev/rmt/0m
exec_parallel        = 0
archive_function     = save | copy_delete_save | double_save_delete | ...
volume_archive       = (<DBSID>A01, <DBSID>A02, ...)
tape_size_arch       = 6000M
tape_address_arch    = [/dev/rmt/1mn]
tape_address_rew_arch = [/dev/rmt/1m]
...

```

Figure 51: Profile of BR*Tools

To configure the behavior of the SAP tools, you can edit the profile with a text editor. If you then start a tool without command options, the values in the initialization profile are used. If a parameter value is not specified in the profile, the SAP tool uses the default value for the parameter.

If you use BRBACKUP or BRARCHIVE with command options, these override the corresponding values in the initialization profile.

The following are the most important parameters for configuring BRBACKUP and BRARCHIVE:

backup_mode

This parameter determines the scope of the backup activity, meaning which part of the database or which directory will be backed up. The two types of a complete database backup are determined through value `all` (whole backup) or `full` (full backup). Partial backup can be indicated, for example, through a tablespace name, file IDs, or a path to a directory. For incremental backups, use `incr`. Value `ora_dir` stands for Oracle software directory, and `sap_dir` for the SAP system directory.

backup_type

With parameter `backup_type`, you can basically choose between online and offline backup.

backup_dev_type

Parameter `backup_dev_type` specifies the backup medium you want to use, such as `tape` or `disk`, or it points out usage of an external backup program through the interface `BACKINT` (parameter value `util_file` or `util_file_online`).

tape_copy_cmd

Parameter `tape_copy_cmd` contains the copy command to be used to copy files from disk to tape (`cpio`, `dd`, `rman`, and so on). This parameter does not affect raw devices (which are always copied with `DD`), or directories (which are always copied with `CPIO`). The profiles `init<DBSID>.ora` and `init<DBSID>.sap`, as well as log files such as summary log and detailed log, are always written with `CPIO` on tape.

disk_copy_cmd

Parameter `disk_copy_cmd` provides the copy command to be used to copy files to local disks. Value `copy` corresponds to the command `cp` on UNIX and `copy` on Windows.



Hint: With BR*Tools 7.00, database and archive log files can be backed up to a local disk using the Oracle utility `OCOPY`.

expir_period and tape_use_count

Parameters `expir_period` and `tape_use_count` are used for tape management and specify the retention period and the recommended maximum number of times to which a volume can be written.

volume_backup and volume_archive

Parameter `volume_backup` provides names of volumes to be used for backups created with `BRBACKUP`. Correspondingly, `volume_archive` provides names of volumes to be used for backups of offline redo log files created with `BRARCHIVE`. For each of these parameters, if you specify more than one volume, you must separate the names with commas and enclose the list in parentheses.

tape_address*

The parameter values `tape_address` and `tape_address_rew` specify the device addresses (device special files) of the tape drives that you want to use to backup the database (or the restore) with or without rewind. BRARCHIVE uses the same values if the optional parameters `tape_address_arch` and `tape_address_rew_arch` are not defined. You may enter more than one device address as parameter value, which indicates that you will perform a parallel backup to several devices, but the number of device addresses in a pair of “no rewind” / “rewind” parameters must always be the same. BRARCHIVE can use two devices as a maximum. Note that values of all these parameters are operating-system-dependent; the above example is valid for HP-UX.



Hint: Regarding the copy command, it is recommended to use DD instead of the default CPIO for copying data files to tapes, because DD is faster and the required backup time can be reduced significantly. For best performance, specify a block size (for copying between disk and tapes) of at least 64 KB, using parameters `ddflags` and `dd_in_flags`. Contact your hardware manufacturer to get more precise recommendations for your tape device.



Note: For a detailed description of all parameters for SAP tools, see the online help at *SAP Library* → *SAP Database Guide: Oracle*.

Integration of Oracle Recovery Manager (RMAN) into SAP Tools

Oracle Recovery Manager (RMAN) is the default Oracle backup and restore program. Its executable runs in a client process and connects to the database similarly to SQL*Plus. By integrating RMAN into BRBACKUP and BRARCHIVE, SAP has added more flexibility to important backup strategies.

BRBACKUP supports using the Oracle Recovery Manager for backup of database files in two different ways:

- RMAN is able to classify a complete database backup as a **level 0 backup** (full backup), which can serve as basis for level 1 backups (incremental backups).
- Data can be written to tapes (or other backup media) using RMAN instead of operating system tools CPIO or DD.

For a complete backup, these two features can be used independently of each other, meaning that each of the four combinations is possible. However, an incremental backup can only be created when RMAN is used as the program for writing data to backup media.



Caution: If you use Oracle Recovery Manager for writing data to backup media, it must also be used for restore and recovery of missing database files. This is recognized and performed by BRRECOVER automatically. Although this is automatically recognized and applied by the SAP tool BRRECOVER, in cases where BRRECOVER runs into a problem and cannot continue, the restore and recovery must be performed on Oracle level, and the user must be an expert in using RMAN.

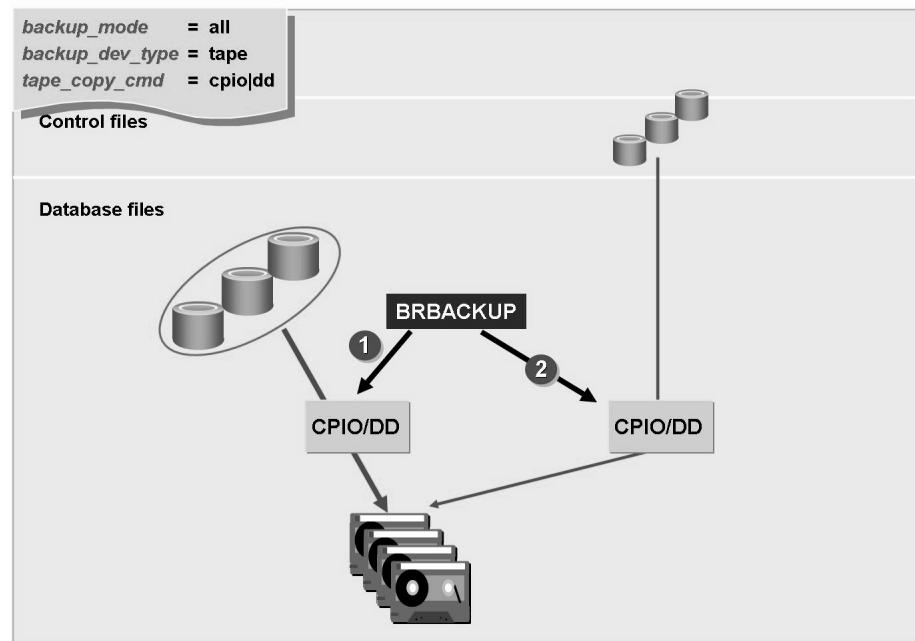


Figure 52: Backups without RMAN

For backups performed without Oracle Recovery Manager (RMAN), then BRBACKUP and BRARCHIVE call CPIO or DD to save database files, including control files to tape.

Full Backup with Operating System Tools and RMAN

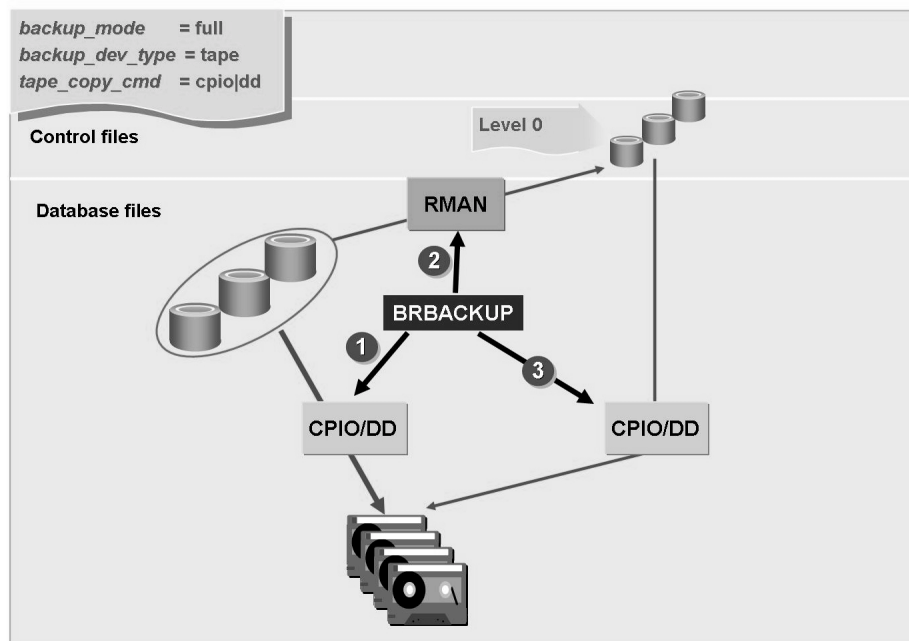


Figure 53: Full backup with RMAN

To perform a full backup, set `backup_mode = full`. This causes RMAN (started by BRBACKUP) to write information about backed up data files to the control file and identify the backup as level 0 backup (full backup).



Hint: When you perform a “native” backup with RMAN directly on Oracle level, RMAN can place information about backups in a separate database called the **recovery catalog**. This catalog is crucial for security because without the recovery catalog data, RMAN cannot recover the database automatically from previous backups. To avoid the need to secure the recovery catalog, the SAP implementation makes RMAN write the backup information only to the control file. To use RMAN directly is not part of SAP strategy for Oracle backups.

To backup files on backup media using BRBACKUP, set parameter `tape_copy_cmd` to either CPIO or DD (when performing a backup to disk, set the parameter `disk_copy_cmd` correspondingly). Data files are then saved to tape with the

command specified. At the end of the backup process, BRBACKUP starts RMAN to write the backup information to the control file, and finally the control file is written to tape with the same operating system tool as before (specified in `tape_copy_cmd`).

Full Backup with RMAN and SBT Library

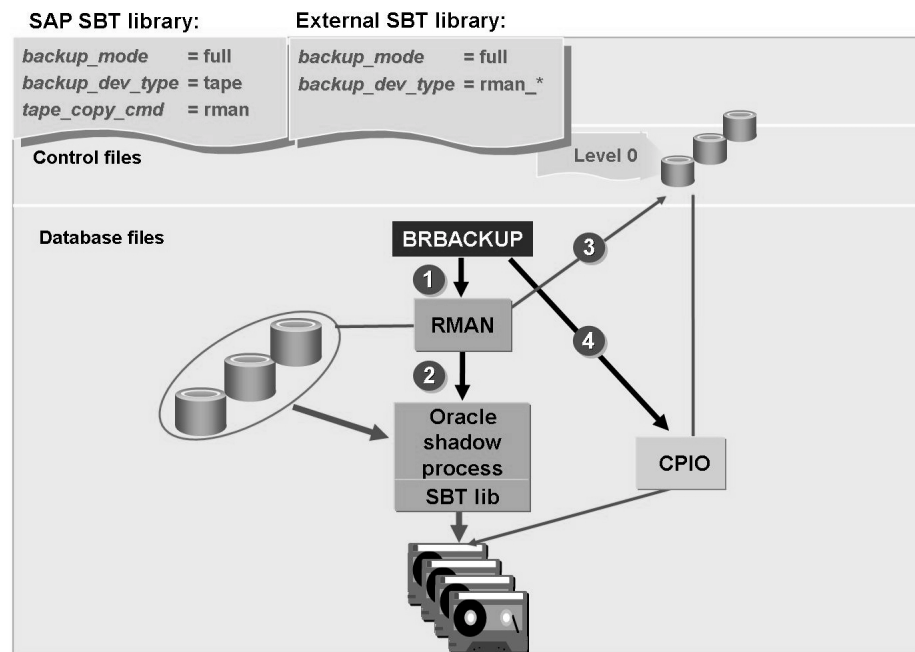


Figure 54: Full Backup with RMAN and SBT Library

To leave control of copying data to backup media to RMAN, set `tape_copy_cmd` (or `disk_copy_cmd`) to the value `rman`. In this case, RMAN is already started by BRBACKUP at the beginning of the backup process, and takes care of backing up the data files. Through its Oracle shadow process, RMAN reads data blocks from the database, checks them for corruption, and filters out those blocks that have never been used (those that are still in the initial status). Used blocks are then written by the shadow process to the backup medium.

In the last phase of the backup process, if `backup_mode` was set to `full`, RMAN writes the backup information to the control file, and CPIO copies the control file to tape.

RMAN can perform backups directly to disk, but not directly to tape. For backups to tape, RMAN uses the System Backup to Tape (SBT) interface provided by Oracle, for which manufacturers of external backup utilities have to provide a library. The SBT library allows data to be backed up to tape directly. Before performing backups using RMAN, you must install the corresponding backup library:

- When you do not use an external backup utility, you must install the SBT library provided by SAP. The SAP SBT library is automatically copied to the directory `/usr/sap/<SAPSID>/SYS/exe/run` during the installation of an SAP system, however it must be made available in the `$ORACLE_HOME/bin` directory. To use the SAP SBT library, set `backup_dev_type` to `tape`, `tape_auto`, or `tape_box` and `tape_copy_cmd = rman` after installing the library.
- When you use an external backup utility, you must install the SBT library of the external backup tool. For information on how to get the SBT library for your external backup, contact the vendor of the external backup tool. Oracle provides a limited single-server version of Legato Networker, including the Legato SBT library, on the first Oracle installation CD. To use the SBT library of an external backup tool, set `backup_dev_type` to `rman_disk` or `rman_util`. Parameter `tape_copy_cmd` is ignored in this case.



Hint: Setting `tape_copy_cmd = rman_disk` or `rman_stage` is possible as of SAP Web AS 6.20. It enables you to perform backups with BR*Tools using RMAN and an external SBT library without having to use the BACKINT interface of the external backup tool.



Note: For more information about installation of the SBT library, see SAP Note 142635.

Using RMAN to copy data to backup media has the following advantages:

- All blocks are checked for block corruption. This ensures that each successful backup contains the database in a consistent state, so an extra verification of the database becomes unnecessary.
- Only used blocks are copied to the backup media. This can reduce the amount of data to be backed up. However, blocks that are empty but have been used before (blocks from dropped tables) are always backed up.
- In a standard online backup, tablespaces are set to backup mode to deal with possible inconsistencies within data blocks. Such an inconsistency can occur, for example, when CPIO or DD performs a copy of an 8 KB Oracle block in smaller operating system units while the block is being overwritten by the database writer process. When a tablespace is in backup mode, whole dirty blocks are copied from the buffer pool to the current redo log instead of writing only modified records there, which can drastically increase the amount of redo log entries. With Oracle Recovery Manager this is not necessary, since the blocks are checked to see if the data is consistent. (More precisely, since each block contains a checksum, RMAN compares the checksum before and after the copy of a block read from disk; if these two checksums are not equal, RMAN reads the block once more.) Consequently, much less redo log information is written during an online backup with RMAN compared to standard online backup.



Hint: As of BR*Tools 7.00, RMAN binary compression can be activated by setting the parameter `rman_compress = no|yes` in `init<DBSID>`.

A whole or partial backup with RMAN (`tape_copy_cmd = rman` or `backup_mode = <object_list>`, `backup_mode = all` or `disk_copy_cmd = rman`;) is possible. All advantages mentioned above apply in this case as well. Obviously, a whole backup is not a level 0 backup and cannot be used as a basis for incremental backups.



Hint: As of SAP Web AS 6.10, BRARCHIVE also supports RMAN backups of offline redo log files with the SAP backup library. The advantage of this process is that the data in the offline redo log files is checked for internal consistency during the RMAN backup. This verification functionality for offline redo log files was missing in older releases.

Save Sets (SAP SBT Library)

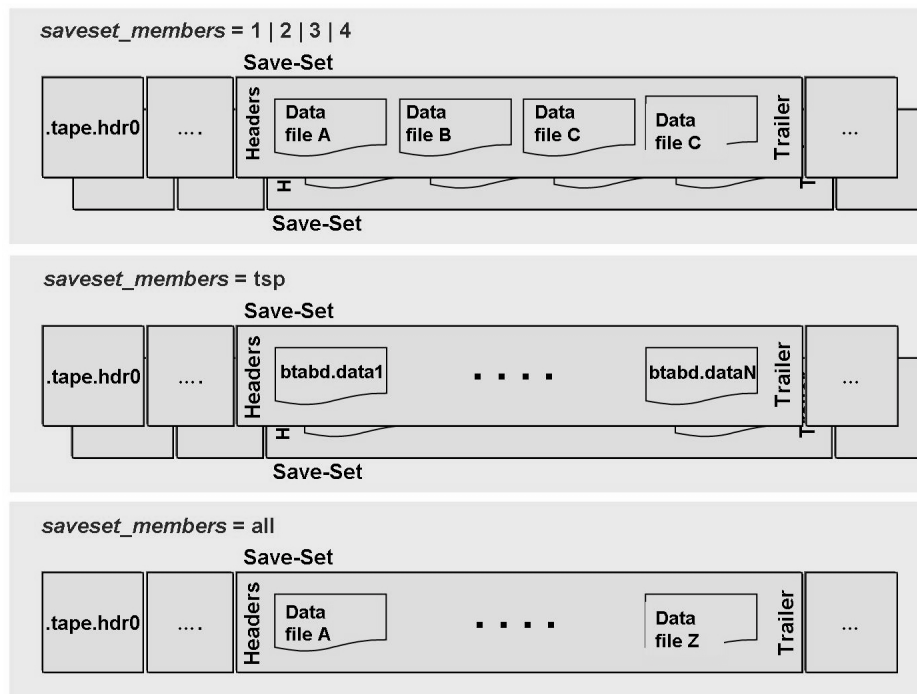


Figure 55: Save Sets

When you back up an Oracle database with the Oracle Recovery Manager, the SAP backup library helps to optimize the utilization of fast tape drives by combining multiple data files in **save sets**. If several data files belong to a save set, RMAN reads the data in parallel from the files. This multiple file access – also known as file multiplexing – maximizes the flow of data to keep tape drives in streaming mode. Through a higher output to tape stations, time required for backup can be reduced.

A save set consists of a header, a trailer, and the blocks of at least one data file. Each save set is treated as an indivisible unit, which must always be stored on one single tape.

The `init<SAPSID>.sap` parameter `saveset_members` determines the (maximum) number of files in a save set. These are possible values of the parameter and their meanings:

- 1, 2, 3, or 4: Number of files to be grouped together to form one save set (default is 1).
- `tsp`: One save set is formed for each tablespace that is to be backed up. The save set contains the data of all data files belonging to a tablespace (as long as they fit onto one tape).
- `all`: Only one save set containing all data files of the database is created, if the tape used is large enough for it.



Caution: Using large save sets can speed up the backup process, but it has the disadvantage that restore/recovery time may be increased. In a situation where only recovery of one (damaged) data file is needed, the complete save set containing this file must be read from the tape. You should therefore try to find out the minimum save set size for your system that guarantees a reasonably fast output to tape devices during backup.

With backups to disks performed with RMAN (`backup_dev_type = disk`, `disk_copy_cmd = rman`), no save sets are formed. Data files are directly copied to disks just like when using CP or DD. In other words: Save sets are created only when an SBT backup library is used for incremental backups.

Preparation Run (SAP-SBT-Library)

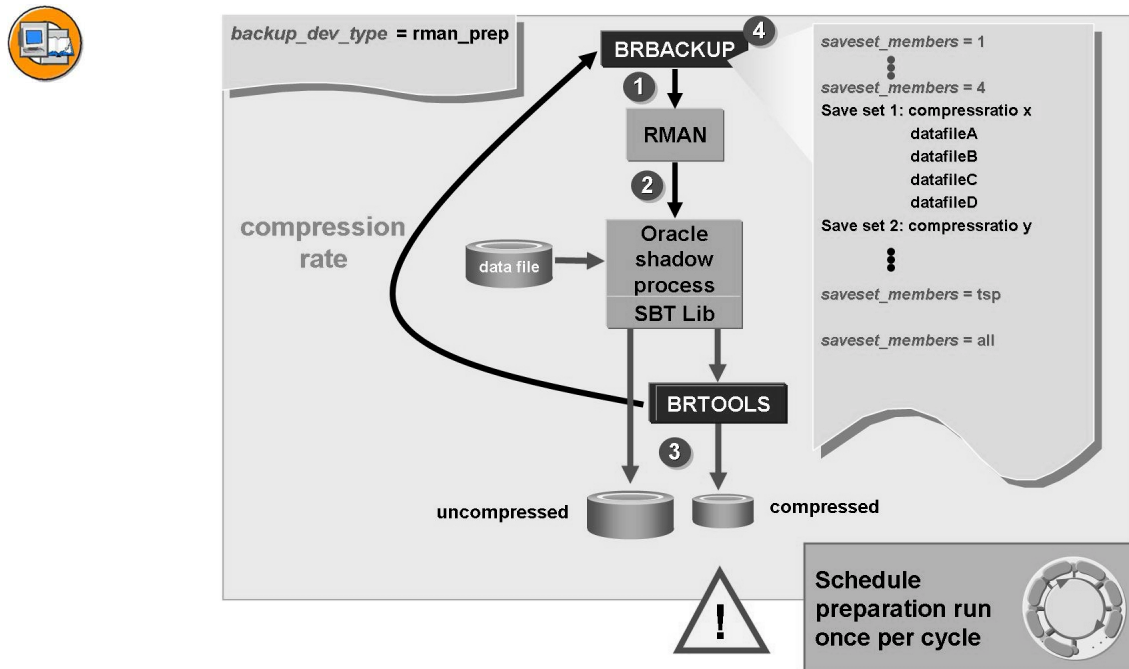


Figure 56: Preparation Run

If a backup with RMAN is supposed to form save sets with more than one member (because you set `save set members` to a value different than 1), or if you use tape stations with hardware compression and you want RMAN to take a compression rate into account when creating save sets with appropriate sizes to match the tape size, you must perform a preparation run to determine the optimal save set distribution of the data files that you want to back up.

In the preparation run, which can be started in the DBA Cockpit or using transaction DB13, action *Prepare for RMAN Backup*, BRBACKUP starts an RMAN backup of every data file to a save set of its own. No backup is really created during this run; the SAP backup library just estimates the compression rate of the save set by letting BRTOOLS compress the file and determine the decompressed and compressed file sizes. The expected compression rate of the save set with one member is then sent to BRBACKUP.

At this point, BRBACKUP determines how data files are allocated to save sets for every possible value of `save set members`, and calculates the compression rate of each save set. The information on the composition of the save sets and the compression rates is stored in the database, and is used during future backup runs.

The allocation of files to save sets cannot be controlled manually. You can only change it (if necessary) through carrying out a new preparation run. Between two preparation runs, save sets that correspond to a specific value of `save_set_members`, remain unchanged, and contain the same files.

If, during a backup, RMAN finds new data files that were not included in the last preparation run (for example, because a data file was added), each of these files is put in its own save set.

It is recommended that you perform a preparation run once per backup cycle and after major database changes, for example, after adding a file to a tablespace, after reorganization, mass data transfer, or after an SAP or database release upgrade.

Incremental Backup (SAP-SBT-Library)

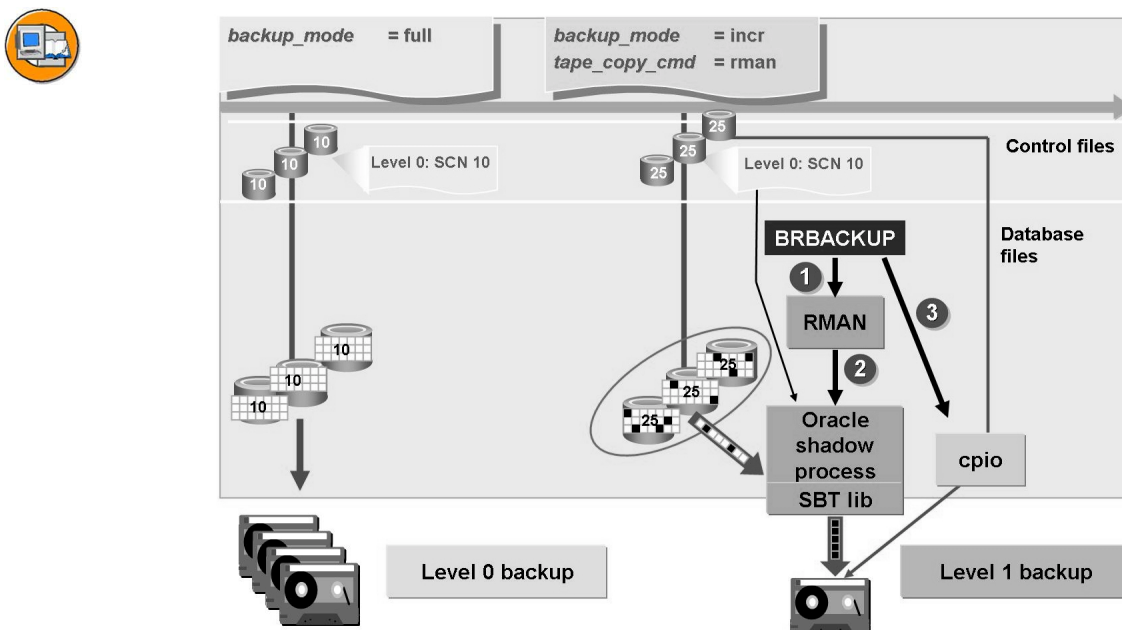


Figure 57: Incremental Backup

An incremental backup, specified with parameter `backup_mode = incr`, is also known as level 1 backup. It is always based on the last level 0 backup (full backup). RMAN reads information about the last level 0 backup from the control files.

An incremental backup is always a backup of the whole database, not of individual data files. Operating system tools cannot be used for writing data to backup media, so for an incremental backup, parameter settings of `tape_copy_cmd` or `disk_copy_cmd` are ignored and implicitly set to `rman`.

After the incremental backup is complete, a control file is saved to tape by CPIO.

In an incremental backup, all blocks of all data files are always read. However, only those blocks that have changed since the last level 0 backup are backed up. An incremental backup can therefore reduce the backup time if the tape stations have less throughput.

With SAP tools, only a “cumulative level 1 backup” is supported as an incremental backup. This means that an incremental backup includes all those blocks that have been already saved during a previous incremental backup (based on the same full backup).

Only one save set (with the extension `. INCR`) is created for an incremental backup. The parameter `saveset_members` is internally set to `all` for an incremental backup run. Since only one save set is created, the backup must fit on one tape. No follow-up tapes may be used.

If data files are added between the last level 0 backup and the level 1 backup, a level 0 backup is performed for these files before the start of the actual level 1 backup. All new data is backed up to one separate save set, which always gets the extension `. FULL`, even if it contains only a part of the database.

Fast incremental RMAN backups

Up to and including Oracle9i, RMAN must read all blocks of a file during an incremental backup—even unchanged files—to determine which blocks have been changed since the last full backup. An incremental backup (level 1) therefore lasts a similar length of time as a full backup (level 0).

The time for incremental backups with RMAN can be reduced considerably as of Oracle 10g with the new RMAN function block change tracking (BCT).

If block change tracking is activated, Oracle keeps a log in the block change tracking file (BCTF) of which data blocks were changed since the last Level 0 RMAN backup. The information from this block change tracking file is then used by RMAN during the incremental backup to read and backup only the changed blocks. There is once block change tracking file for each database. The BCTF is written by the new Oracle background process **CTWR**.

The duration for an incremental backup is roughly proportional to the size of the database if BCT is deactivated, but proportional to the number of changed blocks if BCT is activated. If only a small part of the blocks have been changed between two Level 0 backups, the time saved as a result of BCT is considerable.

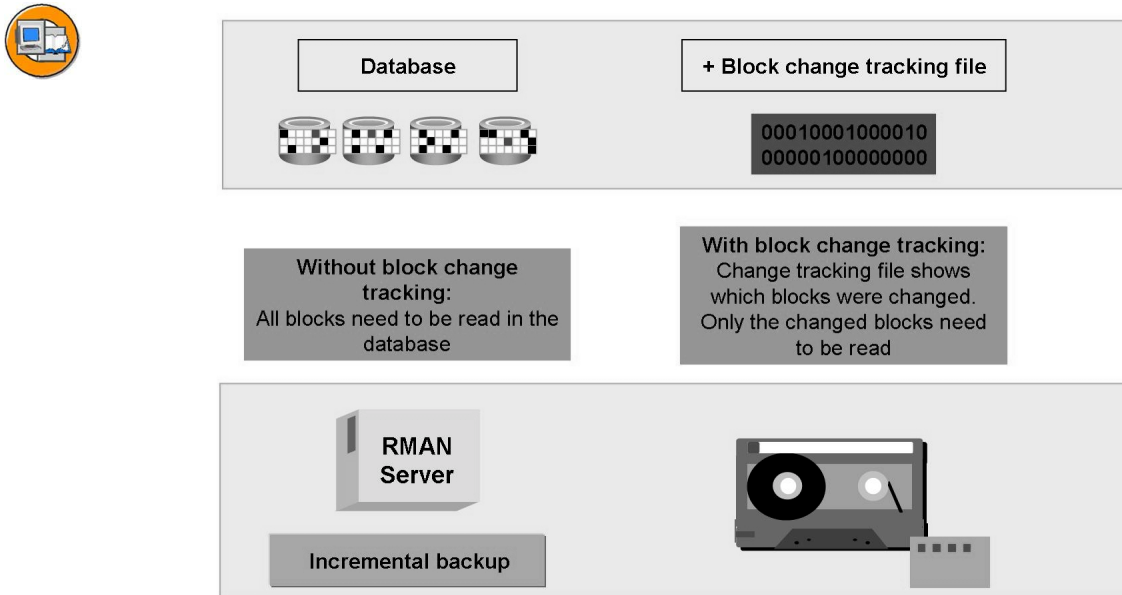


Figure 58: Block Change Tracking

Block change tracking is deactivated by default. When block change tracking is activated for the first time, RMAN must read all blocks per file during the first Level 0 backup because the BCTF does not yet reflect the true, current block status the first time it is created. RMAN can then use the BCTF information in any subsequent incremental backups.



Caution: In the SAP environment, block change tracking is supported as of BR*Tools Release 7.00, patch level 15 for Oracle databases as of Release 10.2.0.2. The BR*Tools configuration does not need to be changed for block change tracking. The BCTF is not backed up, created or managed by BR*Tools.

The block change tracking file has the SAP standard name `bctf<DBSID>.ora`, and is saved in the directory:

- `$ORACLE_HOME/dbs` (Unix)
- `%ORACLE_HOME%\DATABASE` (Windows)

The BCT can be activated and deactivated using the sqlplus command (see SAP Note 964619):

- Activate BCT:
`ALTER DATABASE ENABLE BLOCK CHANGE TRACKING USING FILE '<Filename>' REUSE;`
- Deactivate BCT:
`ALTER DATABASE DISABLE BLOCK CHANGE TRACKING`

Recovery with Incremental Backup

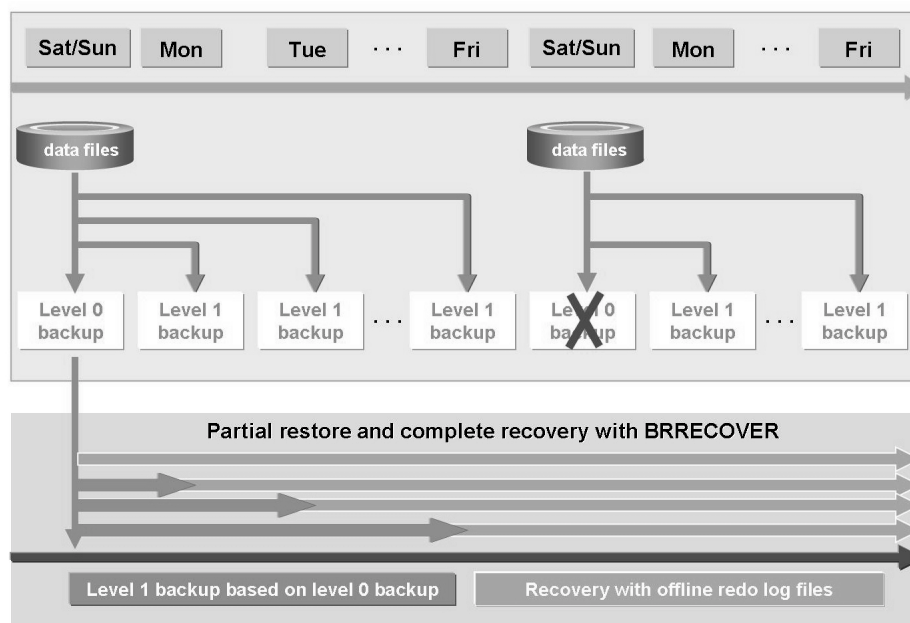


Figure 59: Recovery with Incremental Backup

When a restore/recovery must be carried out (for example, due to a disk crash), a level 1 backup is not sufficient to repair the database. A Level 0 backup of the lost files is always required.

1. The lost files must be recovered from a Level 0 backup.
2. The changed blocks from a level 1 backup (which must be based on the applied level 0 backup) can be imported to the data file.
3. Finally, you must perform a recovery from the time of the level 1 backup.

Since incremental backups created with SAP tools are cumulative, you only need to apply one incremental backup, preferably the latest one.

If no level 1 backup is available for the level 0 backup, perform the recovery based on the last available level 0 backup. This usually takes longer than a recovery using the level 1 backup as a basis.

If you cannot use the last level 0 backup, you must use the previous level 0 backup for restoring the data file. In the second step, only a level 1 backup can then be used, which is based on this level 0 backup. Incremental backups that are based on the damaged full backup cannot be used at all.



Caution: You must perform at least two, preferably four, level 0 backups within one backup cycle. An incremental backup without its corresponding level 0 backup is useless.

External Backup Tools

The SAP tools BRBACKUP, BRARCHIVE, BRRESTORE, and BRRECOVER provide an interface called BACKINT that can be used to access external backup programs. You can only use this interface if the BACKINT interface program is supported by the supplier of the external backup program.

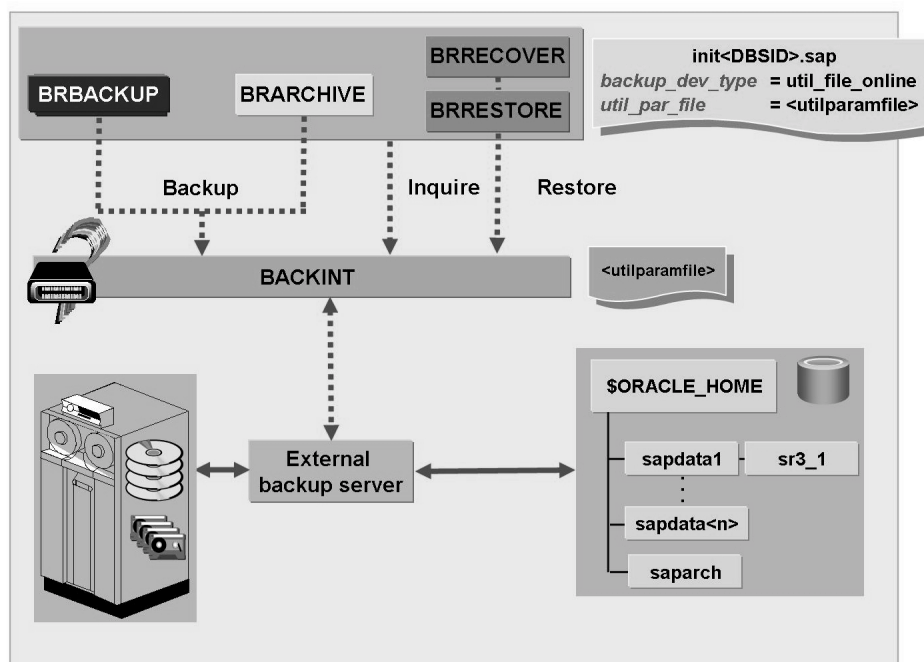


Figure 60: External Backup Tools

By using external backup programs, you gain the following advantages:

- You can use new, manufacturer-specific backup media such as tape robots and magneto-optical media. SAP tools, for example, do not support direct backup to or restore from optical storage media. However, you can use such media with an external backup program and the BACKINT interface.
- You can set up a consistent backup procedure for file systems and databases.
- Client/server backup configuration allows for use of one backup server (including mainframe).

The backups must still be started by SAP tools. This ensures that all actions are logged, and that backups can be monitored using the CCMS. In addition, this allows you to use the restore and recovery features of BRRECOVER.

Backup and restore tasks are performed as follows:

- BRBACKUP, BRARCHIVE, BRRESTORE, and BRRECOVER handle the database.
- The external backup program manages the backup media.
- BRBACKUP or BRARCHIVE uses BACKINT to pass a backup request to the external backup program. This request contains a list of the files for backup. BRRESTORE also uses BACKINT to trigger the external program to restore the requested files. BRRECOVER calls BACKINT while performing disaster recovery.
- The external backup program performs all the backup operations.
- BRBACKUP, BRARCHIVE, BRRESTORE, or BRRECOVER evaluates the confirmation messages of the external backup program.

To configure the BACKINT interface, set parameter `backup_dev_type` to `util_file` or `util_file_online` in file `init<DBSID>.sap`. (If using the second parameter, the tablespace to be backed up is set to the backup mode dynamically, on request from BACKINT.) Second, the value of the parameter `util_par_file` must refer to the configuration file that contains parameters for the external backup utility. For the name and location of the `util_par_file`, refer to the documentation of your external backup utility.

To use external backup tools in combination with RMAN backups, set `backup_dev_type` to `rman_util`, `rman_disk`, or `rman_stage`.



Note: For more information please refer to SAP Note 142635.



Hint: Setting `backup_dev_type` to any `util_file*` or `rman_*` values turns off tape management performed by SAP tools. This means that parameter `tape_copy_cmd` is ignored.



Note: It is permissible to use the BACKINT solution only with certified external backup tools. For more information about SAP partners that support the interface to external backup programs, see <http://service.sap.com/partners>.

Exercise 6: Backup Tools

Exercise Objectives

After completing this exercise, you will be able to:

- Prepare backups to local tapes using tape compression

Business Example

You perform backups on local tapes using hardware compression. As preparation for the backups, you want to determine the compression rate for backups with and without RMAN.

Task:

Determine the compression rate for backups with and without RMAN.

1. Determine the compression rate of backups performed without RMAN.
2. Determine the compression rate of backups performed with RMAN.

Solution 6: Backup Tools

Task:

Determine the compression rate for backups with and without RMAN.

1. Determine the compression rate of backups performed without RMAN.

- a) Start BRGUI or BRTOOLS and choose *Backup and database copy* → *Additional functions* → *Update of compression rates*.

```
BR0657I Input menu 26 - please check/enter input values
```

```
-----
BRBACKUP options for determination of compression rates
```

```

1 - BRBACKUP profile (profile) ..... [initT99.sap]
2 - Database user/password (user) ... [/]
3 - BRBACKUP run type (type) ..... [online]
4 ~ Files for compression (mode) .... [all]
5 - Confirmation mode (confirm) ..... [yes]
6 - Query mode (query) ..... [no]
7 - Parallel execution (execute) .... [0]
8 - Additional output (output) ..... [no]
9 - Message language (language) ..... [E]
10 - BRBACKUP command line (command) . [-p initT99.sap -k only -t
    online -m all-e 0 -l E]
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
```

```
BR0662I Enter your choice:
```

- b) The compression rate is displayed at the end of the output:

```
...
```

```
BR0063I 7 of 7 files processed - 678.500 MB of 678.500 MB done
```

```
BR0204I Percentage done: 100.00%, estimated end time: 15:31
```

```
BR0001I *****
```

```
BR0115I Compression rate for all files 8.2199:1
```

```
BR0056I End of database backup: bdwrkiee.cmb 2007-11-26 15.31.47
```

```
BR0280I BRBACKUP time stamp: 2007-11-26 15.31.48
```

```
BR0052I BRBACKUP completed successfully
```

2. Determine the compression rate of backups performed with RMAN.

Continued on next page

- a) Start BRGUI or BRTOOLS and choose *Backup and database copy* → *Additional functions* → *Preparation of RMAN backups*.
- b) The compression rate is displayed at the end of the output for each number of save sets:

BR0001I *****

BR0527I Save sets with 1 file:

Saveset	Size	Rate	Compressed	Name
1	132907008	2.1508:1	61794175	G:\ORACLE\ T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1
2	21037056	2.3721:1	8868721	G:\ORACLE\ T99\SAPDATA2\UNDO_1\UNDO.DATA1
3	31457280	6.6715:1	4715198	G:\ORACLE\ T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1
4	2097152	6.3520:1	330158	G:\ORACLE\ T99\SAPDATA3\T99_1\T99.DATA1
5	131072	117.9766:1	1111	G:\ORACLE\ T99\SAPDATA3\T99USR_1\T99USR.DATA1

BR0527I Save sets with 2 files:

Saveset	Size	Rate	Compressed	Name
1	164364288	2.4713:1	66509373	G:\ORACLE\ T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1 G:\ORACLE\ T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1
2	23134208	2.5149:1	9198879	G:\ORACLE\ T99\SAPDATA3\T99_1\T99.DATA1 G:\ORACLE\ T99\SAPDATA2\UNDO_1\UNDO.DATA1
3	131072	117.9766:1	1111	G:\ORACLE\ T99\SAPDATA3\T99USR_1\T99USR.DATA1

BR0527I Save sets with 3 files:

Saveset	Size	Rate	Compressed	Name
1	166461440	2.4905:1	66839531	G:\ORACLE\

Continued on next page

```

T99\SAPDATA3\T99_1\T99.DATA1
G:\ORACLE\
T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1
G:\ORACLE\
T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1
2      21168128      2.3865:1      8869832      G:\ORACLE\
T99\SAPDATA3\T99USR_1\T99USR.DATA1
G:\ORACLE\
T99\SAPDATA2\UNDO_1\UNDO.DATA1

```

BR0527I Save sets with 4 files:

Saveset	Size	Rate	Compressed	Name
1	187498496	2.4766:1	75708252	G:\ORACLE\
T99\SAPDATA3\T99_1\T99.DATA1				
				G:\ORACLE\
T99\SAPDATA2\UNDO_1\UNDO.DATA1				
				G:\ORACLE\
T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1				
				G:\ORACLE\
T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1				
2	131072	117.9766:1	1111	G:\ORACLE\
T99\SAPDATA3\T99USR_1\T99USR.DATA1				

BR0527I Save sets with 'tsp' files:

Saveset	Size	Rate	Compressed	Name
1	2097152	6.3520:1	330158	G:\ORACLE\
T99\SAPDATA3\T99_1\T99.DATA1				
2	131072	117.9766:1	1111	G:\ORACLE\
T99\SAPDATA3\T99USR_1\T99USR.DATA1				
3	21037056	2.3721:1	8868721	G:\ORACLE\
T99\SAPDATA2\UNDO_1\UNDO.DATA1				
4	31457280	6.6715:1	4715198	G:\ORACLE\
T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1				
5	132907008	2.1508:1	61794175	G:\ORACLE\
T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1				

BR0527I Save sets with 'all' files:

Continued on next page

Saveset	Size	Rate	Compressed	Name
1	187629568	2.4783:1	75709363	G:\ORACLE\ T99\SAPDATA3\T99_1\T99.DATA1
				G:\ORACLE\ T99\SAPDATA3\T99USR_1\T99USR.DATA1
				G:\ORACLE\ T99\SAPDATA2\UNDO_1\UNDO.DATA1
				G:\ORACLE\ T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1
				G:\ORACLE\ T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1

BR0280I BRBACKUP time stamp: 2007-11-26 14.50.13

BR0533I Uncataloging save sets created by RMAN...

BR0522I 5 of 5 files/save sets processed by RMAN

BR0280I BRBACKUP time stamp: 2007-11-26 14.50.20

BR0534I Save sets created by RMAN uncataloged successfully

BR0056I End of database backup: bdwrkekr.rmp 2007-11-26 14.50.20

BR0280I BRBACKUP time stamp: 2007-11-26 14.50.21

BR0053I BRBACKUP completed successfully with warnings



Lesson Summary

You should now be able to:

- Identify the different SAP tools for backup, restore, and recovery
- Explain the concept of Oracle's Recovery Manager (RMAN)
- Customize the SAP tools

Lesson: Appendix: Tape Management with BR*Tools

Lesson Overview

BR*Tools also offer functions for tape management, similar to other backup utilities. Therefore, tape management with BR*Tools is also covered in this lesson.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe tape management with BR*Tools
- Initialize and manage backup tapes with BR*Tools

Business Example

After defining your backup strategy and providing the necessary number of tapes, you want to perform the backups. To do so, you must first customize the parameters of BR*Tools for backup and restore and initialize your tapes.

Tape Management

To facilitate the management of tapes used for backups of your Oracle database, BRBACKUP and BRARCHIVE offer a tape management system that:

- Helps you find and correctly use the tapes that are necessary to perform a backup
- Helps you find the appropriate tapes when you need to restore and recover your database
- Provides for tape protection so that tapes are not accidentally overwritten

Tape Pools

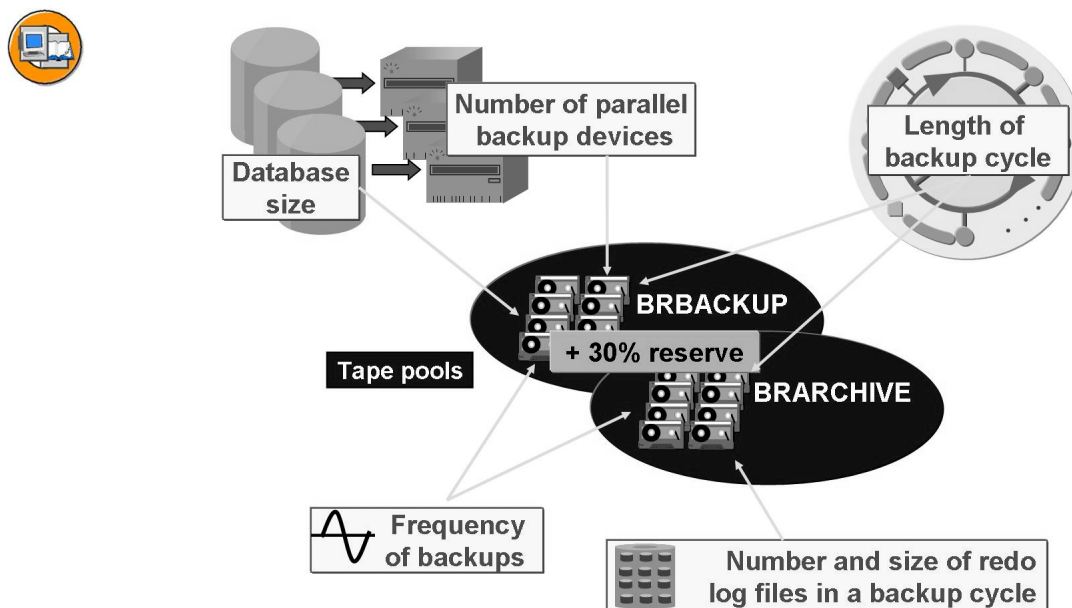


Figure 61: Tape Pools

For Oracle administration in an SAP system, two different tools are used for backups: BRBACKUP for database backups, and BRARCHIVE for offline redo log file backups. Consequently, one separate pool of tapes is required for database backups, and another one for offline redo log file backups. You must ensure that enough tapes are provided in each tape pool to span the entire backup cycle. Backup tapes from each pool can be reused at the end of the backup cycle.



Hint: If backups performed with BRBACKUP and BRARCHIVE are written directly to tape, at least one tape is needed for each run of any of these tools. In other words, you cannot save the backup of two backup runs to the same tape.

The number of tapes you need for database backups is the product of two factors:

1. Number of tapes needed for one complete database backup. This number depends on the size of your database and on the capacity of the tapes you use.
2. Number of backups per cycle. This depends on the length of your backup cycle and the frequency of database backup operations.

The number of tapes you need for **backing up offline redo logs** depends on:

1. The average number and the size of the redo log files created in a backup cycle (which, in turn, depends on the length of your backup cycle and on the activities in your database).
2. The storage capacity of the tapes you use

The number of tapes for backing up offline redo log files may also be influenced by the frequency of these backup operations in relation to database activities. If the redo information created between two offline redo log backups does not fill one tape, you will need more tapes for the cycle.



Hint: The default actions of the DB13 templates within the DBA Cockpit *Whole database offline + redo log backup* and *Whole database online + redo log backup* will backup the data files and offline redo log files in one run. This strategy saves tapes, as no extra tape pool for BRARCHIVE is necessary (assuming that data files and offline redo logs fit on one tape).

In addition to the number of tapes you need based on your backup strategy, you should have a reserve of 30% more tapes in each tape pool. This is useful in the case of database growth, exceptionally high redo log volume caused by additional activities in the database, or if additional backups need to be performed.

Initializing Tapes

A prerequisite for using this tape management system is two pools of initialized tapes. You must initialize one pool of tapes for BRBACKUP and another one for BRARCHIVE. Tapes that are initialized by BRBACKUP should not be used by BRARCHIVE, and vice versa. Tapes that are not initialized at all are rejected as well.



└ Profile init<DBSID>.sap contains the tape names

```
...
volume_backup = (<DBSID>B01,<DBSID>B02,...)
volume_archive = (<DBSID>A01,<DBSID>A02,...)
...
```

└ Initialize new tapes, non-SAP tapes, or locked tapes:

```
brbackup -i force or brarchive -i force
```

└ Rename non-locked tapes:

```
brbackup -i -v <tape name> or brarchive -i -v <tape name>
```

└ Label containing the tape name is written to the tape as the first file

.tape.hdr0

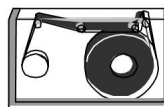


Figure 62: Initializing Tapes

During tape initialization, an SAP-specific label is written on the tape as the first file (.tape.hdr0), containing the tape name (volume name). You can either specify the tape name explicitly, or BRBACKUP/BRARCHIVE will automatically select the tape names from the pool of names defined in the configuration file init<DBSID>.sap by parameters volume_backup and volume_archive. We suggest the following naming convention for your tapes:

- <DBSID>B01, <DBSID>B02, ... , <DBSID>Bxx for BRBACKUP
- <DBSID>A01, <DBSID>A02, ... , <DBSID>Axx for BRARCHIVE

To initialize tapes, start BRTOOLS or BRGUI and choose *Backup and database copy* → *Additional functions* → *Initialization of BRBACKUP tape volumes* or *Initialization of BRARCHIVE tape volumes*. As an alternative, you can start BRBACKUP or BRARCHIVE with the option -i | -initialize and specify all necessary options on the command line.

When started with BRTOOLS, the following menu is displayed:

```
BR0657I Input menu 32 - please check/enter input values
```

```
-----
Options for initialization of BRBACKUP tape volumes
```

```
1 - BRBACKUP profile (profile) ..... [initT99.sap]
2 - Initialization type (initialize) . [rename]
3 ~ Number of volumes (number) ..... []
```

```
4 - Confirmation mode (confirm) ..... [yes]
5 - Message language (language) ..... [E]
6 ~ Tape volume names (volume) ..... [T99B01]
7 - BRBACKUP command line (command) .. [-p initT99.sap -i -l E -v T99B01]
```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

The options are:

initialize

The default initialization type is `rename`. Renaming a tape is only possible for tapes that were previously initialized and whose retention period has expired. In all other cases, initialization must be performed with initialization type `force`. To avoid overwriting a used tape with the `force` type, first check the label using initialization type `show`.

number

If empty, all not-yet-initialized tapes are initialized. Set this to initialize only a certain number of tapes.

volume

If empty, the tapes specified in the `init<DBSID>.sap` parameter `volume_backup` and `volume_archive` respectively, are initialized. To initialize specific tapes, enter their volume names here, separated by commas.

Tape Label Contents

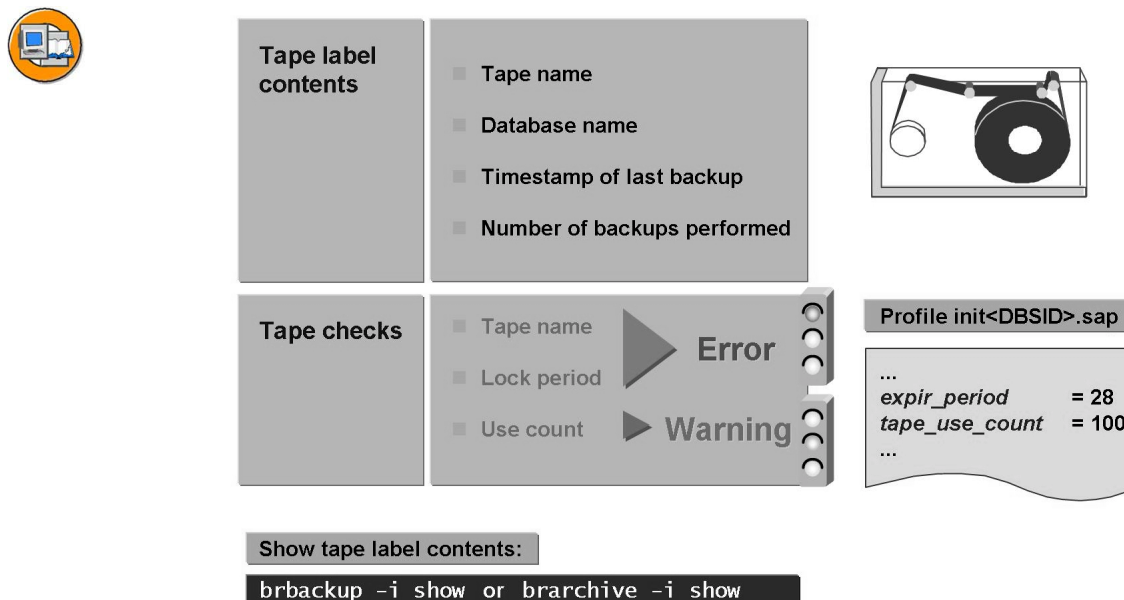


Figure 63: Tape Label Contents

After an initialized tape has been used by BRBACKUP or BRARCHIVE, the tape label contains the following information:

- The name of the tape
- The name of the database for which the backup was performed
- The timestamp of the last backup recorded on the tape
- The number of backups performed with this tape

By default, BRBACKUP and BRARCHIVE read the tape label before they start writing to a tape in order to check:

- The tape name
- Whether the tape is locked (the configured expiration period – the number of days specified in the `init<DBSID>.sap` parameter `expir_period` that must have passed before the volume can be used again – has not ended yet)
- The number of times the tape has already been used

If the tape name is wrong or if the tape is locked, an error is reported and the tape is not used. If the tape is used more often than the value set in parameter `tape_use_count` in file `init<DBSID>.sap`, a warning is generated but the tape is used.

The expiration period always expires at midnight of the last day of the lock. If you set an expiration period of 0 days, the volume is not locked at all; it can be overwritten on the same day.

Tape Checks

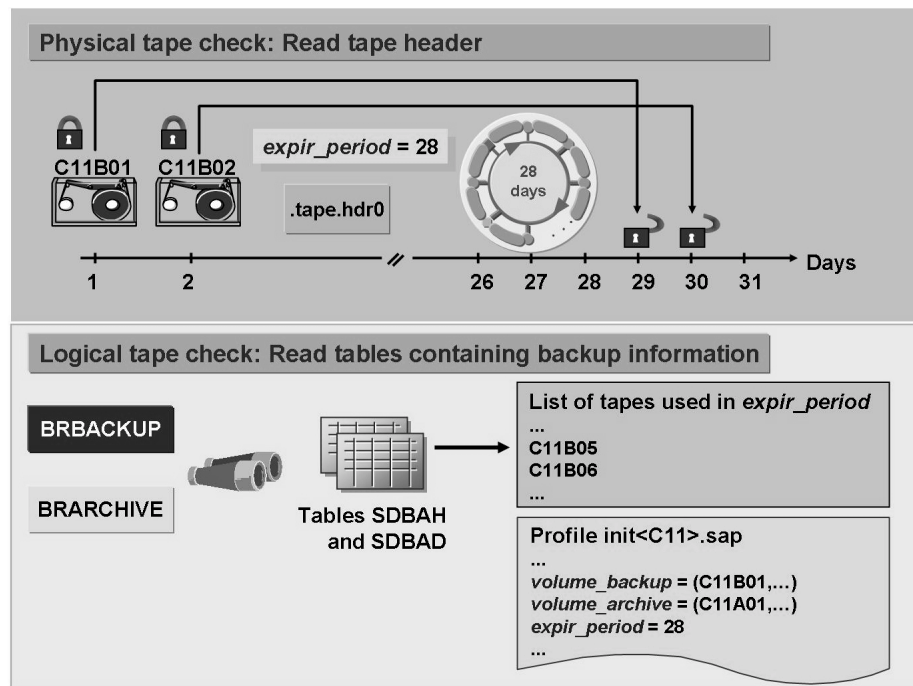


Figure 64: Tape Checks

At the beginning of a backup, BRBACKUP/BRARCHIVE writes the timestamp to the header file on tape. Additionally, when a database file has been backed up successfully, information about it (including timestamp) is written to special backup

log files (summary log and detailed log), and also to the database tables *SDBAH* and *SDBAD*. That is why BRBACKUP and BRARCHIVE can use two different methods for checking that a tape is not locked:

- The **physical lock** check is derived from the tape label. The timestamp of the last backup found in the tape label and the parameter `expir_period` found in `init<DBSID>.sap` determine whether the tape can be reused. If the number of days passed since the tape was last used is less than value of parameter `expir_period`, the tape is physically locked.
- The **logical lock** check is derived from the timestamp written to the tables *SDBAH* and *SDBAD*. To find which tapes can be used for the next backup and distinguish them from those that are still locked, BRBACKUP connects to the database and searches tables *SDBAH* and *SDBAD* for the tapes that were used in the lock period. These tapes cannot be used for the next backup; they are locked logically. The tape that follows the last used tape in the parameter `volume_backup`, and is not contained in the list of tapes used in the lock period, is selected for the next backup. After the last volume on the list is used, the first volume on the list is requested again.

The logical lock check for the offline redo log file backups is performed by BRARCHIVE using information from the summary log. Therefore, offline redo log files can be backed up even when the database is not available.

Under certain circumstances, discrepancies may occur between the physical and logical locks.

At the beginning of a backup, the volume label is written to the tape. If the backup is then terminated before the first database file could be written to the tape, the volume is locked physically but not logically. At next backup run, it is selected from the list (`volume backup` or `volume archive`) but rejected when the physical volume label check takes place. You must therefore reinitialize the volume with the same name in order to cancel the physical lock. Because the expiration period for this tape is not over yet, you must use initialization type `force` to do so.

If you reinitialize a volume before the expiration period ends (using the initialization type `force`), then this volume is no longer locked physically. However, it will not be selected automatically for the next backups as long as it remains locked logically. If you want to use this volume before the logical lock has expired, you can switch off the automatic tape selection temporarily by inserting the tape and performing the next backup for a volume named `SCRATCH`.

Tape Selection

SAP tools provide three procedures for selecting a tape for a backup:

- Automatic tape selection by BRBACKUP or BRARCHIVE
- Manual tape selection by the operator
- Tape selection by an external tool

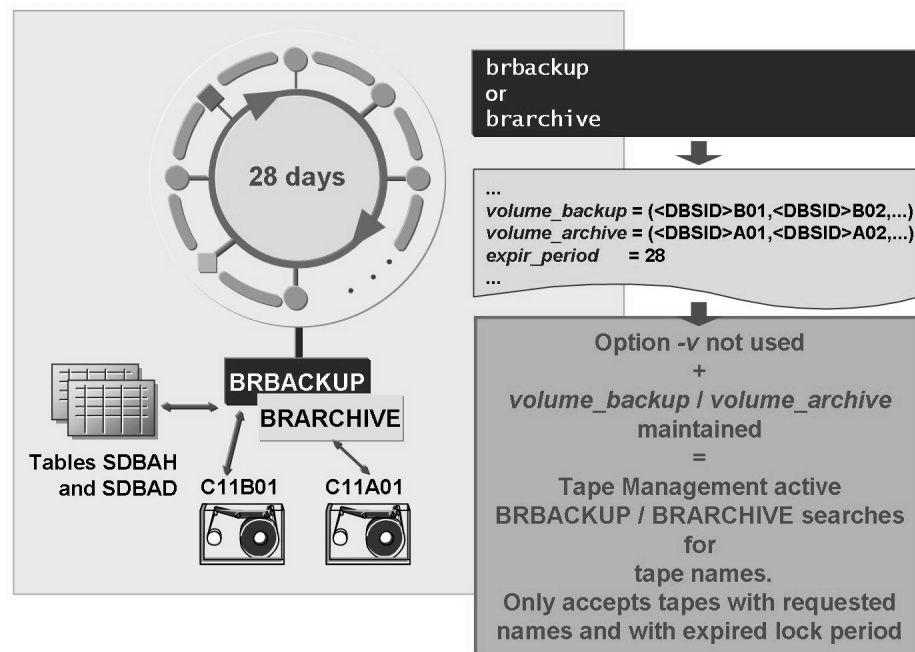


Figure 65: Automatic Tape Selection

If you want BRBACKUP or BRARCHIVE to select the tape(s) to be used for the next backup run automatically, you must:

- Define the parameters `volume_backup` and `volume_archive` in profile `init<DBSID>.sap`.
- Not specify a volume when calling or scheduling the backup program

BRBACKUP/BRARCHIVE then performs the logical lock check and requests the next unlocked tape in the order defined by parameter `volume_backup` or `volume_archive`. The mounted tape is checked physically as well. If the operator does not mount the requested tape, the program aborts with error.

To check which tape will be automatically selected by a specific backup, select a backup in the DBA Cockpit or transaction DB13 and display the action details by double-clicking the selection. On the operating system level, you can check this with `brbackup|brarchive -q| -query [check]` to check which tape will be automatically selected on the next backup. Use parameter `check` to also perform a physical tape check.

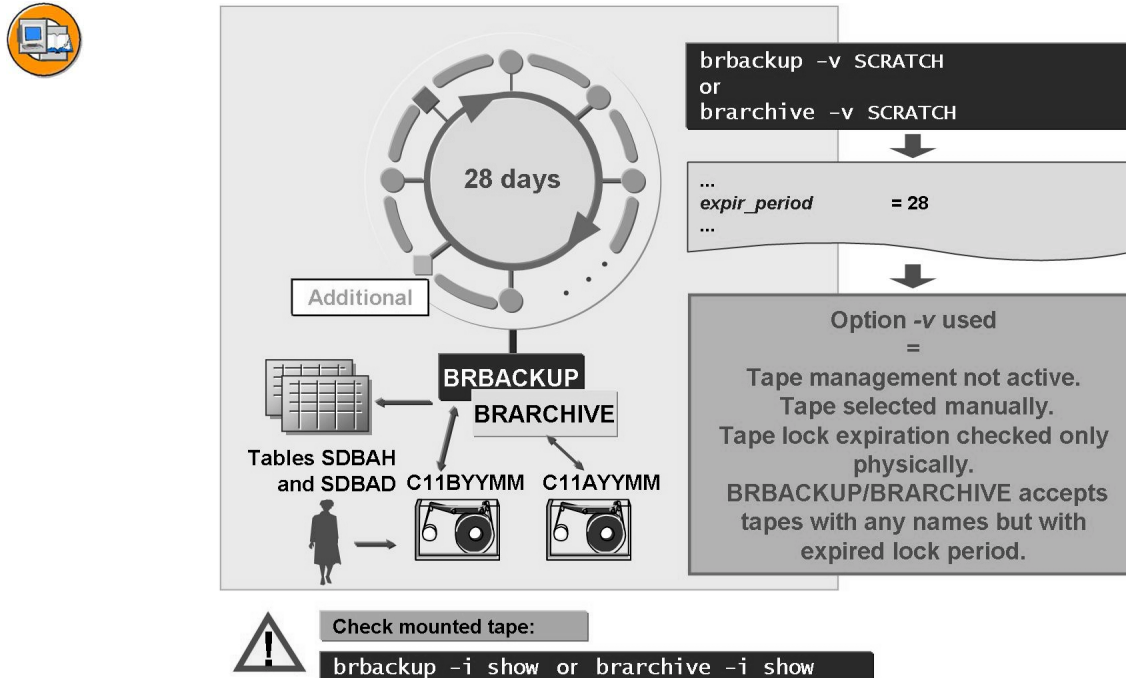


Figure 66: Manual Selection of Tape with Any Name

To allow for a backup on a volume with any name, specify the volume `scratch` when starting a backup from BRTOOLS or BRGUI, menu *Backup and database copy* → *Database backup* and *Backup and database copy* → *Archivelog backup* respectively, or use the option `-v| -volume scratch` for BRBACKUP/BRARCHIVE.

Performing a backup specifying the symbolic name `scratch` switches off automatic tape management. BRBACKUP/BRARCHIVE performs just the physical lock check, and accepts any initialized tape with an expired lock period. The operator must check that the proper tape is mounted and used for this backup. Even if by accident the “wrong” tape is mounted, it will be used (as long as it is not locked), and the old data on it will be overwritten.

Use this method, for example, for creation of an additional month-end backup if you do not want this backup to be performed on your tape pool tapes.

You can also initialize some tapes with the symbolic volume name *scratch*. These scratch tapes can be used, for example, to replace a defect tape in your tape pool. When automatic tape selection is used (meaning no tape volume was specified with the backup), a scratch tape will be automatically renamed to the volume name of the tape that was requested.



Hint: Do not mix up the two scratch options:

- A backup performed specifying the volume *scratch* will accept any initialized tape which is not physically locked.
- A backup performed not specifying any volume name (automatic tape selection) will accept a tape that was initialized with volume name *scratch* and rename it to the requested volume name.

External or Manual Tape Selection

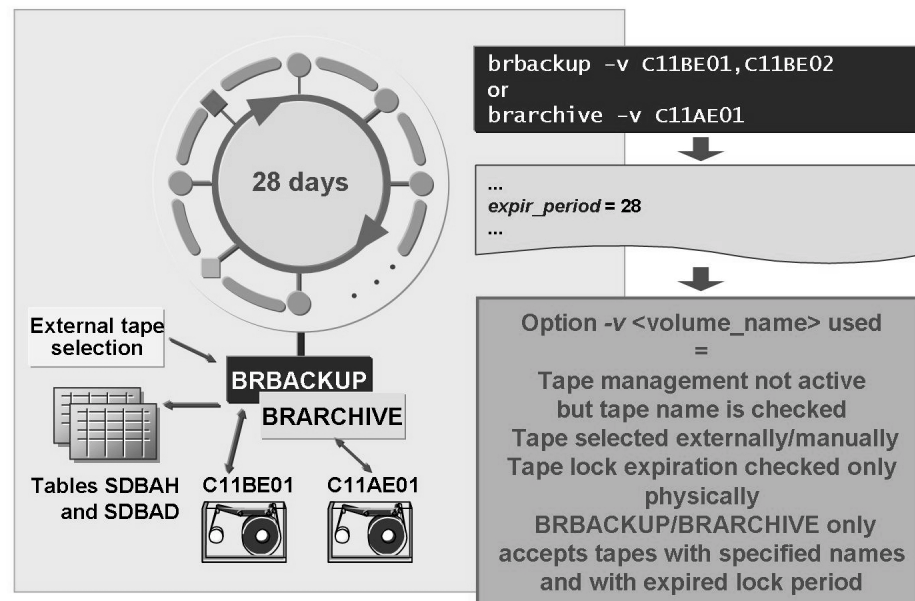


Figure 67: External/Manual Tape Selection with Specified Name

To explicitly specify the tape(s) to be used by BRBACKUP or BRARCHIVE, specify the volume(s) to be used with the option *volume* when starting a backup from BRTOOLS or BRGUI, menu *Backup and database copy* → *Database backup* and *Backup and database copy* → *Archivelog backup* respectively, or use the option

-v|-volume <volume> for BRBACKUP/BRARCHIVE. Specifying a volume always deactivates the automatic tape selection, but the physical tape check is performed and locked tapes are rejected.

Tape Layout

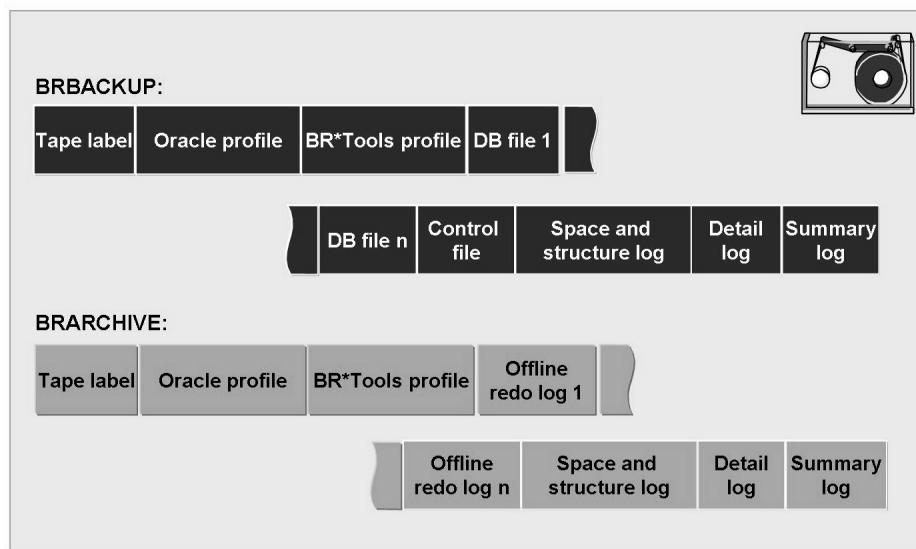


Figure 68: Tape Layout

The files written to tapes by BRBACKUP and BRARCHIVE are:

- .tape.hdr0: Tape label
- init<DBSID>.ora: Database configuration file (Oracle profile)
- init<DBSID>.sap: BR*Tools configuration file
- space<DBSID>.log: Information about the creation, extension, or reorganization of tablespaces or tables (on disk located in directory sapreorg)
- struc<DBSID>.log: History of database structure changes (located in directory sapreorg)
- <action_ID>.<function-ID>: Detail log of BRBACKUP/BRARCHIVE: complete output of the BRBACKUP or BRARCHIVE run (located in directory sapbackup or saparch)
- back<DBSID>.log: Summary log of BRBACKUP: list of all backups started with BRBACKUP (located in directory sapbackup)
- arch<DBSID>.log: Summary log of BRARCHIVE: list of all offline redo log files backed up by BRARCHIVE (located in directory saparch)

Configuring the Correct Tape Size

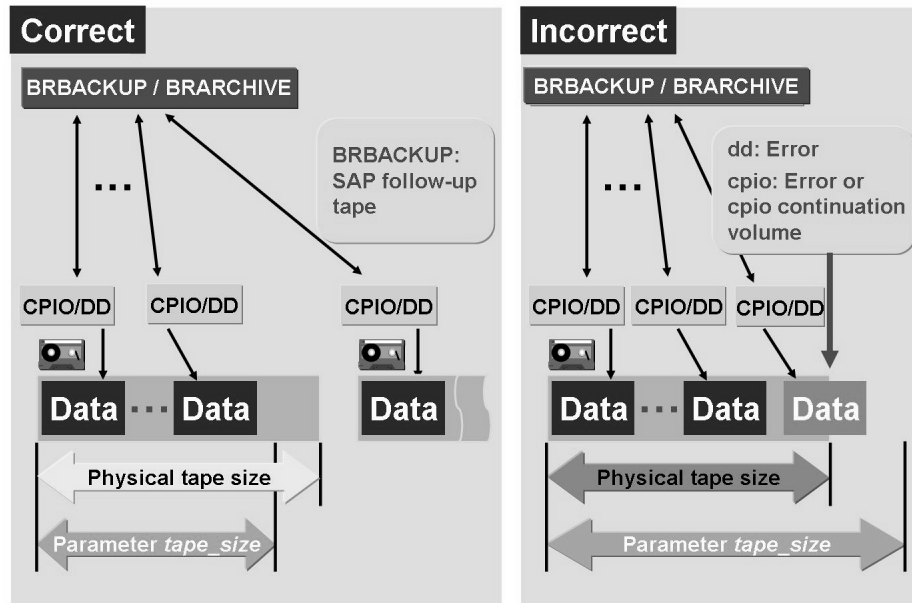


Figure 69: Initialization Parameter *tape_size*

The programs BRBACKUP and BRARCHIVE get the information about the memory capacity of the tapes to be used from the parameter *tape_size* (or *tape_size_arch*). Parameter *tape_size* logically defines the memory capacity in gigabytes (GB), megabytes (MB) or kilobytes (KB) for the tapes that will be used for backups with BRBACKUP. Its value is also valid for tapes used by BRARCHIVE if the corresponding BRARCHIVE parameter *tape_size_arch* is not set. At the beginning of a backup, BRBACKUP/BRARCHIVE determines the data volume to be backed up and plans the distribution of this data over initialized SAP tapes with help of the specified parameter value. Files are never split; they are backed up to tape in one piece. The same is true for files on raw devices.



Caution: The largest file and the largest raw device volume for backup may not be larger than the value specified in *tape_size* (after compression, when applicable).

The value of *tape_size* / *tape_size_arch* should be slightly smaller than the physical tape capacity because a few more megabytes of space are needed (compared to the total size of data to be backed up) for backing up init files and backup log

files, as well as for writing CPIO file headers. This additional space is not taken into consideration when the total space needed is calculated. To be on a safe side, allow for a 10% safety margin when setting `tape_size/tape_size_arch`.

When all files are copied to a tape according to the plan, no matter how much space remains free on the tape, BRBACKUP asks for an SAP follow-up tape. After the tape has been made available, BRBACKUP continues backing up data.

In contrast to BRBACKUP, BRARCHIVE does not have its own management of follow-up tapes. During an offline redo log file backup, the maximum number of offline redo log files that can fit on one tape (as defined by `tape_size` or `tape_size_arch`) is backed up. An SAP follow-up tape is not used. When the tape is full, you must start a new BRARCHIVE run to write to the next volume.

If the value for `tape_size` is too large, too many files may be planned for a copy to one tape. The copy program (CPIO or DD) will then reach the physical end of the tape while copying a file that does not fit onto the tape anymore. The consequences depend on the copy program and the type of backup:

- The copy program DD always generates an error message when it reaches the end of the tape. The error message depends on the operating system. In Windows the message reads “Physical End of tape has been reached”, and in UNIX “I/O-Error.” The backup process terminates with an error.
- During a serial database (or offline redo log file) backup, CPIO requests a CPIO (not a SAP tool) continuation volume and the backup process continues.



Caution: Although the database backup terminates successfully, problems may arise during a restore from this database backup, since SAP tools do not request the CPIO continuation volume directly.

- During a parallel database (or offline redo log backup), CPIO stops with an error message, and the entire backup process terminates with an error.

Reaching the physical end of tape should generally be avoided.

Hardware Compression and Tape Size

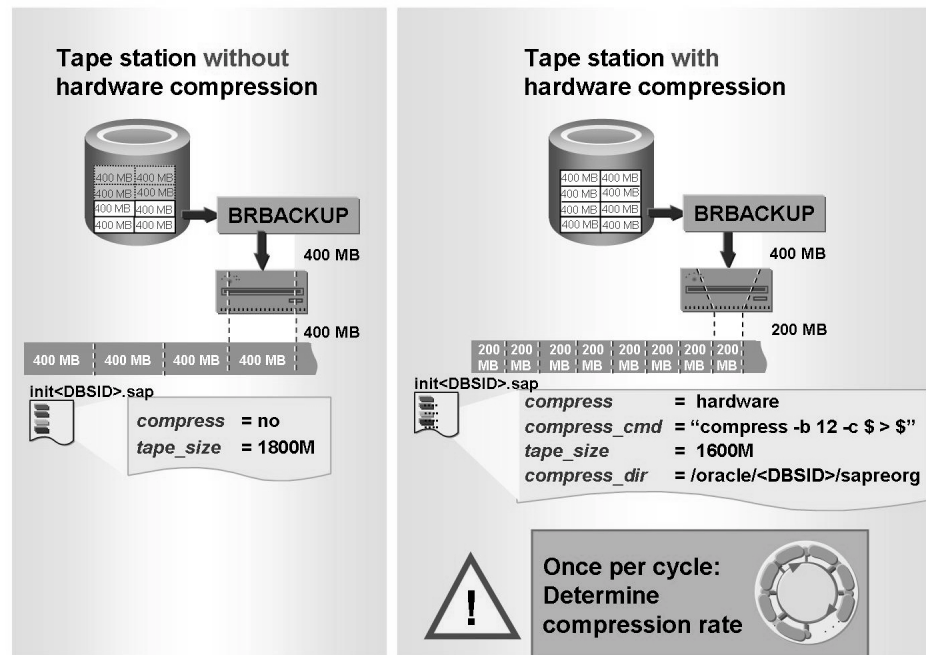


Figure 70: Hardware Compression

With the help of initialization parameter `compress`, you can determine whether files will be compressed during backup, or whether BRBACKUP/BRARCHIVE takes hardware compression into consideration when planning distribution of files over tapes.

compress=no

Software compression is not used. Hardware compression may be used depending on the backup device, but BRBACKUP/BRARCHIVE is not aware of it.

compress=yes

Software compression is used. In this case, you must also set the `compress` command in the parameter `compress_cmd`. The parameter value depends on the operating system; examples are within the comments of `init<DBSID>.sap`. Furthermore, you specify the directory in which compression is to be performed in parameter `compress_dir`.

compress=hardware

This parameter can be set when tape units that support hardware compression are used. Of course, just setting this parameter does not activate hardware compression. It is merely information for BRBACKUP to use the current compression rates (as for software compression) when calculating how much data will fit onto one tape. You must also configure your backup device accordingly.

If you use software or hardware compression for files, the parameter `tape_size` specifies the total size of the files that will fit on one tape after compression. The corresponding space needed to store compressed files on tapes is calculated by BRBACKUP with help of the current compression rates. Only when using the correct compression rates of the database files can the backup tool properly determine the quantity of data to be saved on one tape after the compression, ensure that the specified tape size is not exceeded, and make sure that the database files are correctly distributed across the tapes.

When backup devices with hardware compression are used, BRBACKUP can only estimate the quantity of data that can be written to a volume because these tools cannot directly determine the compression rates for hardware compression (tape stations do not report a compression rate). BRBACKUP uses the software compression rates as an estimate for hardware compression rates. BRARCHIVE always assumes compression rate 1:1 (no compression) for offline redo log files.

Before the first database backup using tape devices with hardware compression, you must start a compression run to determine the compression rates. In the DBA Cockpit or transaction DB13, start the action *compress database*, or execute `brbackup -k only`. This call does not actually start a backup; it only determines the compression rates. The database files are merely compressed (not saved) and the determined compression rates are stored in a database table (*SDBAD*) and in a detail log of BRBACKUP.



Caution: Repeat this activity of updating compression rates at least once per backup cycle / once a month, and additionally after a reorganization or after loading of a large amount of data.



Hint: To determine the compression rates as close to the actual hardware compression rates as possible, set the parameter `compress_cmd` to `compress -b 12 -c $ > $` on UNIX platforms, and to `mkszip -l 0 -c $ > $` on Windows (see SAP Note 19909).

Since the actual space needed for storing compressed files on tapes may differ from the value calculated by BRBACKUP with help of compression rate estimates, it is crucial to set the value of the parameter `tape_size` to an even smaller value than without compression (as an additional safety margin) to prevent the problem of reaching physical end of tape during backup.



Hint: If you want to use hardware compression, minimize the risk for reaching physical end of tape, and circumvent the need for updating the compression rates, set the parameter `compress` to `no` and `tape_size` to twice the physical size of your tapes. This works in most cases because the compression rates are almost always at least 2:1.

If parameter `exec_parallel` is set to 0 during compression rate determination, one process per logical volume is triggered to determine the compression rate. If you set `exec_parallel` to a positive value smaller than the number of logical volumes, then the number of processes required to determine the compression rate is limited to the number indicated by the parameter value. This reduces the CPU load on the database server.



Lesson Summary

You should now be able to:

- Describe tape management with BR*Tools
- Initialize and manage backup tapes with BR*Tools

Related Information

- Use a URL or a cross-reference tag to point out additional information that the participants may find useful, such as Web sites or White Papers. Delete this if it is not relevant.

Lesson: Performing Backups

Lesson Overview

This lesson explains how to perform backups using BR*Tools.



Lesson Objectives

After completing this lesson, you will be able to:

- Perform online, offline, and partial backups
- Perform RMAN backups, including incremental backups
- Create backups of archived redo log files

Business Example

While your daily backups run fine, you have learned about different backup types and scenarios and you want to test other scenarios.

Introduction

The primary task of BR*Tools with respect to data security is to back up all business data. However, your backup strategy must include backing up all objects, including the operating system, files associated with the SAP system (SAP executables, interfaces, and archiving objects), and files in the Oracle software directories. These objects are usually backed up on operating system level (consider creating such a backup at least once per backup cycle).

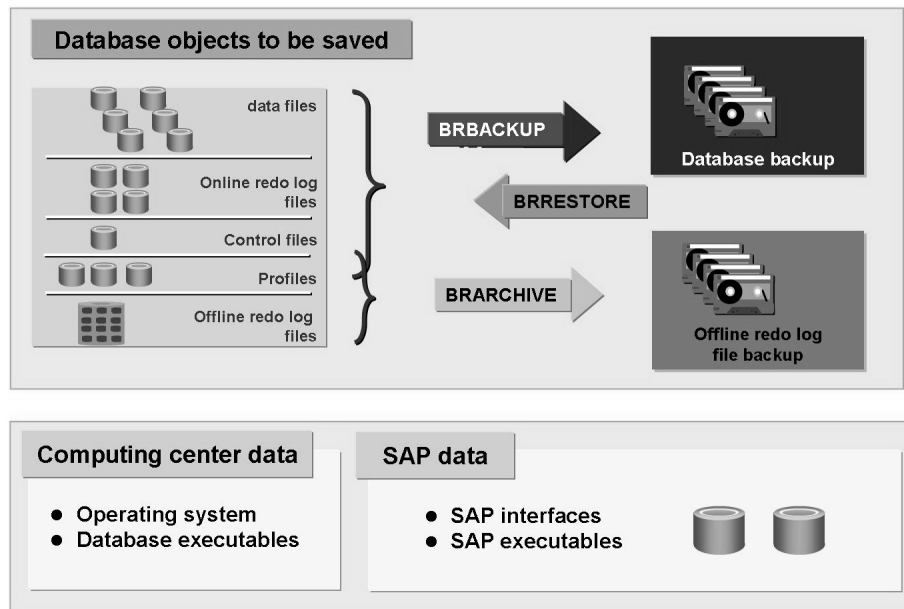


Figure 71: Objects for Backup

In addition to the selected files, BRBACKUP always backs up the control file, the profile, and specific log files. A complete offline backup also backs up online redo log files. BRARCHIVE primarily saves offline redo log files in the backup medium, along with the profiles and log files.

Phases of a Database Backup

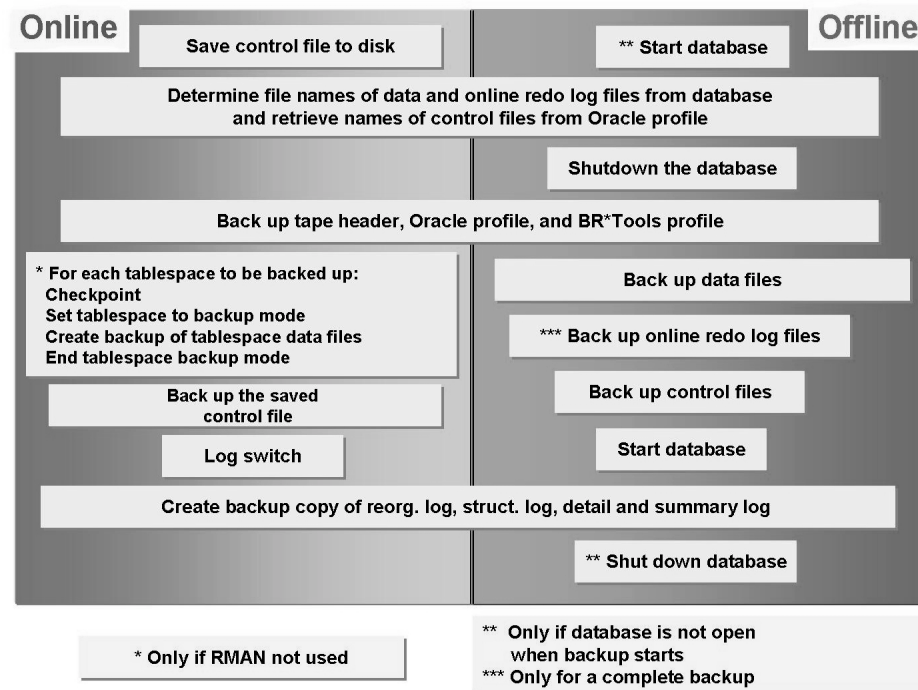


Figure 72: Phases of a Database Backup

Some steps of offline and online backup procedures are identical, but not all.

Both types of backup write a tape header (label) at the start of a backup to tape, and read the header at the end of the backup. By reading the header, BRBACKUP checks whether the tape could be written correctly.

Both types of backup also write the profiles, the control file, and some log files to the backup medium.

In case of an online backup, the control file cannot be backed up during normal database operation. Therefore, at the start of the backup, a consistent copy of this control file is made to disk. This copy is backed up to tape after all files have been backed up.

In a standard online backup, without using the Oracle Recovery Manager and an SBT library, backup of each tablespace is performed in **backup mode**. The precise procedure is as follows.

The tablespace enters the backup mode (Oracle statement: ALTER TABLESPACE BEGIN BACKUP). This has three consequences:

- A checkpoint on tablespace level is triggered before the backup of this tablespace starts, and the database writer copies all corresponding dirty blocks of this tablespace to the data files.
- The system change number (SCN) corresponding to the tablespace checkpoint remains “frozen” in the headers of all tablespace files until the end of backup mode.
- Every data change performed in this tablespace is logged in the redo log on data block level instead of data record level; not only the redo information for a modified record is written into the redo log, but the whole 8 KB block containing this record. The reason for this less granular logging is the possibility that CPIO or DD may copy an Oracle block to the backup medium in smaller units than 8 KB while the block is being changed by the database writer, which can lead to inconsistencies within such a block. When the backup is later used for restoring the tablespace, all blocks written to the redo log during the backup mode overwrite their “suspicious” versions saved directly in the backup of the data files. The “frozen” SCN informs Oracle where the part of the redo log that contains the blocks of the corresponding tablespace starts.

When the last data block of the tablespace has been backed up, the tablespace is reset, and the backup mode ends (ALTER TABLESPACE END BACKUP).

Online backup with Oracle Recovery Manager does not set tablespaces into backup mode because RMAN guarantees that each data block is copied to the backup medium in a consistent state.

At the end of an online backup, BRBACKUP performs a redo log switch. You can then carry out a backup of offline redo log files and be sure that all redo information written during your online backup will be there. This is very important for data security; since activities (data changes) are allowed in the database during an online

backup, it does not contain a consistent snapshot of the database data. When copied back to disk during restore of the database, the data can only be made consistent if the corresponding redo information is available.



Hint: Do not shut down the database in an extra step as a preparation for an offline backup. Since BRBACKUP needs to read tables *SDBAH* and *SDBAD* (containing a log of backups) and some Oracle dictionary views at the beginning of each backup procedure, it opens the database when it is closed. BRBACKUP then shuts down the database automatically before it starts backing up data files, and it opens it again at the end to write backup protocol into *SDBAH* and *SDBAD*. BRBACKUP leaves the database as it was found, so it will shut it down if it was not open at the beginning.

The online redo log files are only included in the backup during a complete offline backup.

Integration of SAP Backup Tools

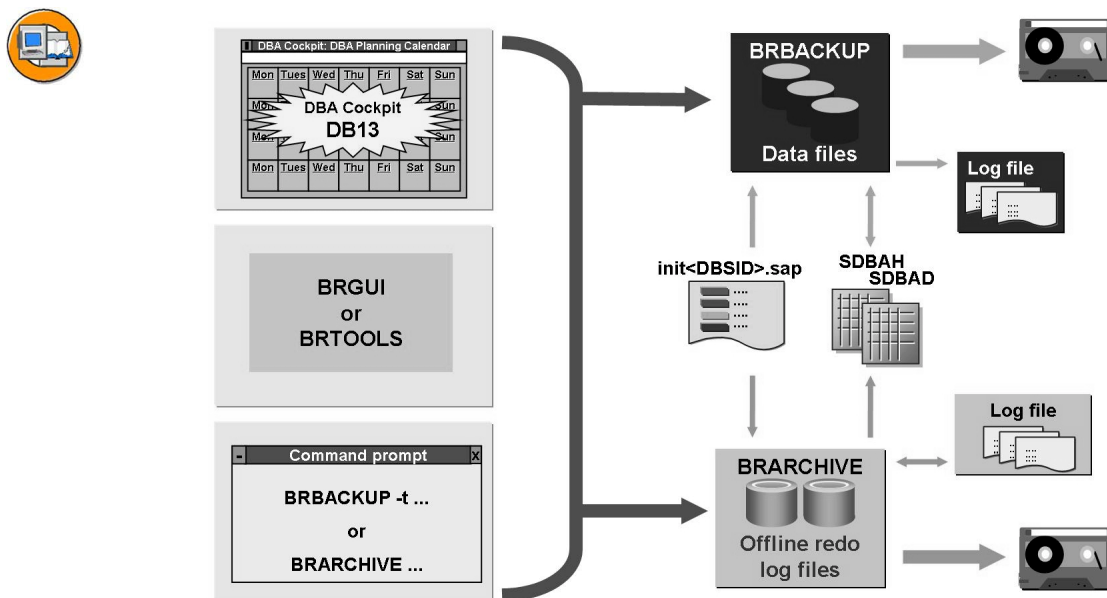


Figure 73: Integration of SAP Backup Tools

There are several interfaces through which you can start or schedule a database backup or an offline redo log backup. No matter which one you use, you will always start BRBACKUP or BRARCHIVE, and these two tools always log backup actions in database tables *SDBAH* and *SDBAD*, as well as in their own summary log and a

detail log per action. For internal tape management, BRBACKUP determines the required tapes from tables *SDBAH* and *SDBAD*, while BRARCHIVE does so on the basis of its summary log.

For processing, BRBACKUP and BRARCHIVE read values of configuration parameters from the BR*Tools profile `init<DBSID>.sap`. Values found there override default values set in the code. However, parameter values specified at command line, in a job definition, or interactively in a menu have the highest priority.



Hint: Selecting parameter values for a BRBACKUP/BRARCHIVE run does not change the values in the profile. To change a parameter value in the profile, you must use an operating system editor. There is no SAP transaction for the maintenance of this profile.

Creating database backups

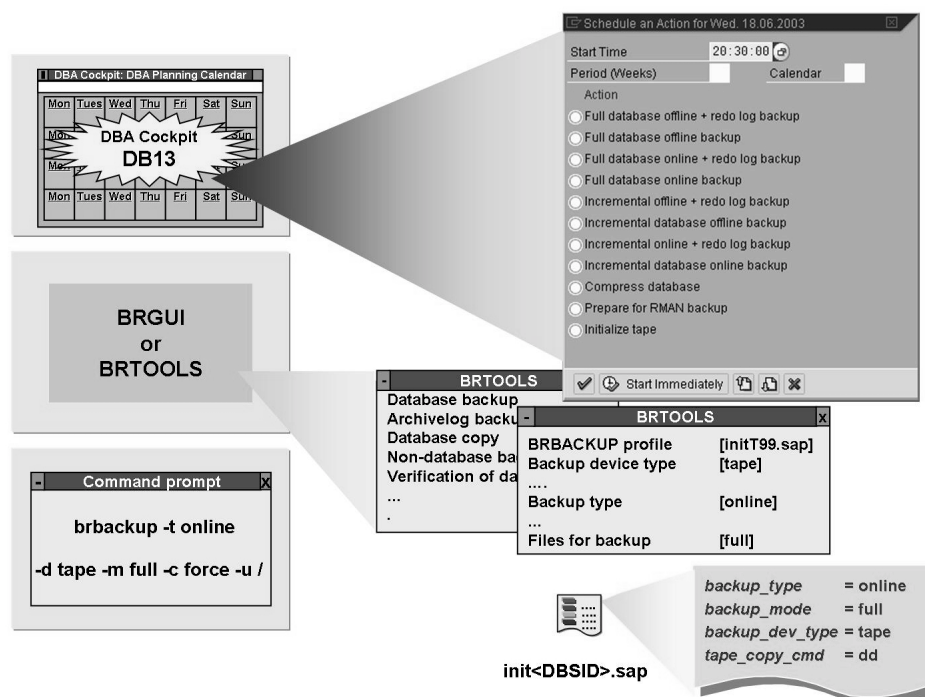


Figure 74: Scheduling and Performing Database Backups

At SAP level, in the DBA Planning Calendar within the DBA Cockpit (transaction DBACOCKPIT) or transaction DB13, you can schedule various types of BRBACKUP and BRARCHIVE jobs. This is the recommended method for scheduling of all

periodic database and offline redo log backups in a backup strategy. The jobs are created as common SAP background jobs (which call BRBACKUP or BRARCHIVE on operating system level), so that the scheduling can be monitored in SM37. Selecting a certain type of backup in DB13 corresponds to particular choice of configuration parameter values.

In a special case where you are not allowed to or do not want to access the SAP system through SAP GUI, you can schedule BRBACKUP and BRARCHIVE jobs on operating system level using their command line options and operating system scheduling (UNIX: CRON; Windows: AT or graphical task scheduler).

For all further database backups (one-time actions and exceptional cases), you can use BRGUI or BRTOOLS and start a backup run interactively. Choose *Backup and database copy* from the main menu, then select the backup function you want to perform. Each selection allows you to specify relevant program options for the BRBACKUP/BRARCHIVE action as needed.

Performing Backups Using BRTOOLS or BRGUI

In most cases, backups are scheduled from the DBA Cockpit or transaction DB13.

To perform a backup of the database with BRTOOLS or BRGUI, choose *Backup and database copy* → *Database backup*. This will display the following menu:



Performing Backups Using BR*Tools

```
BR0657I Input menu 15 - please check/enter input values
```

```
-----
BRBACKUP main options for backup and database copy
```

```
1 - BRBACKUP profile (profile) ..... [initT99.sap]
2 - Backup device type (device) ..... [disk]
3 # Tape volumes for backup (volume) . []
4 # BACKINT/Mount profile (parfile) .. []
5 - Database user/password (user) .... [/]
6 - Backup type (type) ..... [offline]
7 # Disk backup for backup (backup) .. [no]
8 # Delete disk backup (delete) ..... [no]
9 ~ Files for backup (mode) ..... [all]
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
```

```
-----
BR0662I Enter your choice:
```

You have the following choices (the preset values in the menu were read from `init<DBSID>.sap`):

Backup Device Type (device)

Specify the device to which the backup should be performed. The main device types are:

- `tape` to perform a backup to tape. Device types `tape_auto` and `tape_box` support tape autoloaders and jukeboxes.
- `disk` to perform backups to a local directory specified by the `backup_root_dir` parameter in `init<DBSID>.sap`.
- `stage` to perform backup to a remote directory specified by the `stage_root_dir` parameter in `init<DBSID>.sap`.
- `util_file` and `util_file_online` for backups using external backup tools with BACKINT.
- `rman_disk` to perform backups using RMAN and an external SBT library. Profiles, control files, and log files are saved to disk. Or use `rman_util` for RMAN with external SBT library. Profiles, control files, and log files are saved via BACKINT.

Tape volumes for backup (volume)

If the backup device type is `tape`, `tape_auto`, or `tape_box`, you can optionally specify the volume.

Backup Type (type)

Main backup types are `online` and `offline`. When performing an offline backup, the database must be shut down. To force the database to shut down even when an SAP system is connected, use `offline_force`.

Back up disk backup (backup)

BRBACKUP fully supports a two-phase backup strategy, where first a backup is performed to device `disk` and, in a second step, this disk backup is saved on tape. Select `yes` to backup a previous disk backup to tape and decide with option *Delete disk backup (delete)* if you want to delete the disk backup after successful backup to tape.

Files for backup (mode)

Here, the backup mode (`all` | `full` | `incr`) is specified. For a partial database backup, specify the tablespace(s) to be backed up.

Special Backups and Options

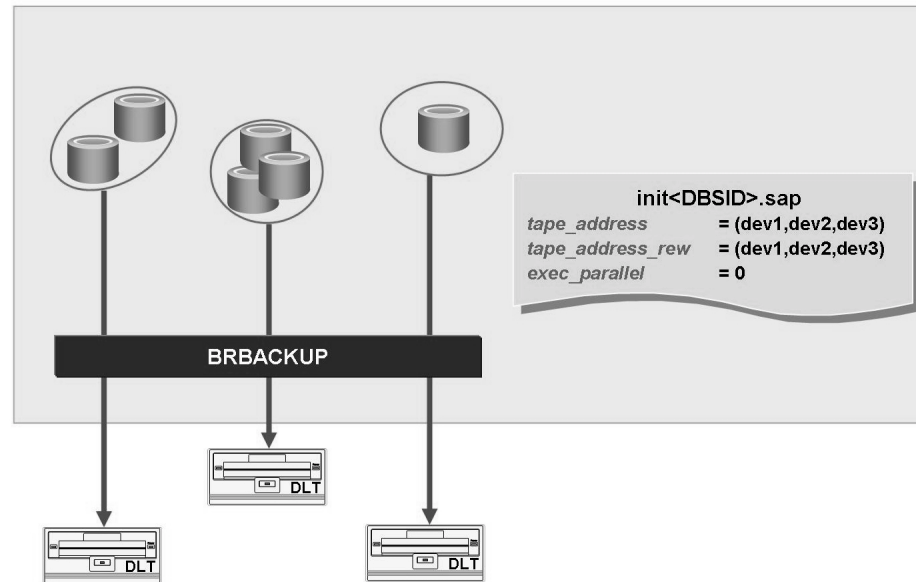


Figure 75: Parallel Database Backup

To reduce the time required to back up and restore the data files and offline redo log files, SAP backup tools support parallel use of several tape stations.

BRBACKUP uses all tape stations defined in parameters `tape_address` and `tape_address_rew` in the profile `init<DBSID>.sap`. Both parameters must contain the same list of tape station addresses (no-rewind and rewind driver), and all used tapes must have the same size.

Database files selected for a backup are distributed across the tapes mounted in the tape stations. To keep backup times to a minimum, the tape capacity should be significantly larger than the total volume of data to be backed up to a tape.

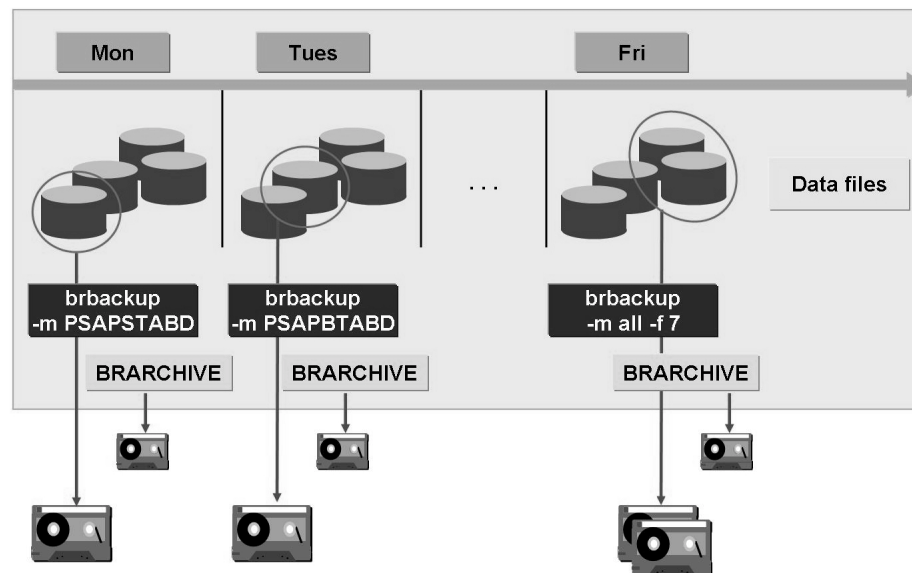


Figure 76: Partial database backups

Should one complete database backup take too long in your production environment, you can split the complete backup into several partial backups. However, the sum of individual backups must cover the entire database in the selected time interval in which you would normally create one complete backup.

Both Oracle and SAP tools support the recovery of data files from different backup runs. For this type of recovery, these tools require all offline and online redo log files generated since the oldest backup of data files.

To ensure that the complete database is backed up within the selected time interval, use BRBACKUP option `-f | -fill <days>`. The corresponding backup run completes the partial backups performed for the previous few days (specified as `<days>`). You must use BRBACKUP directly or from BRTOOLS/BRGUI, as this option is not supported in transaction DB13.



Hint: This procedure can also be used to complete aborted backup runs. In this case, specify the log file associated with the aborted backup run:
`-f | -fill <log_file> | last.`

Verifying backups

Even if a backup is reported as successfully completed, the backup might not be error free. There are two types of verification to check if the backup is good.

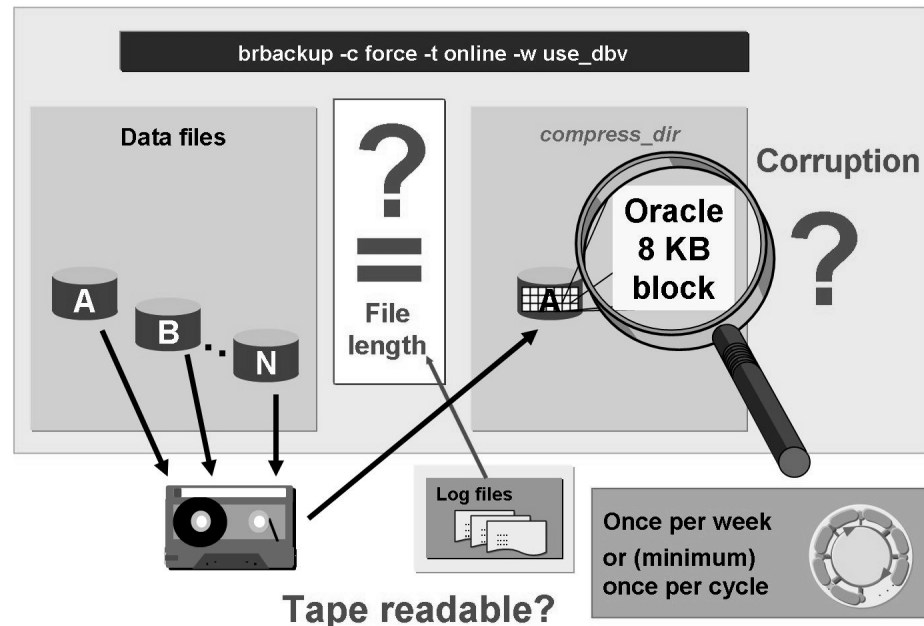


Figure 77: Verifying backups

Tape verification

To verify if a backup is readable, the files are restored file by file and compared with the originals. Depending on the type of backup, different checks are used, from binary compares to just comparing the size when online backups were performed.

Run this check once per week or, at minimum, once per backup cycle.

Block consistency

This type of verification checks the database itself block by block using the Oracle tool DBVERIFY.

Run this check at least once per backup cycle. Whenever you have a bad Oracle block in a data segment, you must be able to restore the corresponding data file from a backup that does not have this bad block.

To perform a backup verification, start BRTOOLS or BRGUI and choose *Backup and database copy* → *Verification of database backup* and *Backup and database copy* → *Verification of archivelog backup* respectively. Select the backup to be verified from the list and select the type of verification in the option *Use DBVERIFY (use_dbv)*:

- `no` means that only tape verification will be performed.
- `yes` means that tape verification is performed **and** database block consistency is checked.



Hint: When verification is started from the command line or from BRTOOLS/BRGUI, you can also perform a database block consistency check without tape verification using option `-w|-verify only_dbv`.

Creating backups of archived redo log files

After a log switch, the Oracle process ARC0 copies the online redo log file that was the current redo log file before the log switch to directory `oraarch` as an offline redo log file. BRARCHIVE copies offline redo log files from this directory to a backup medium.

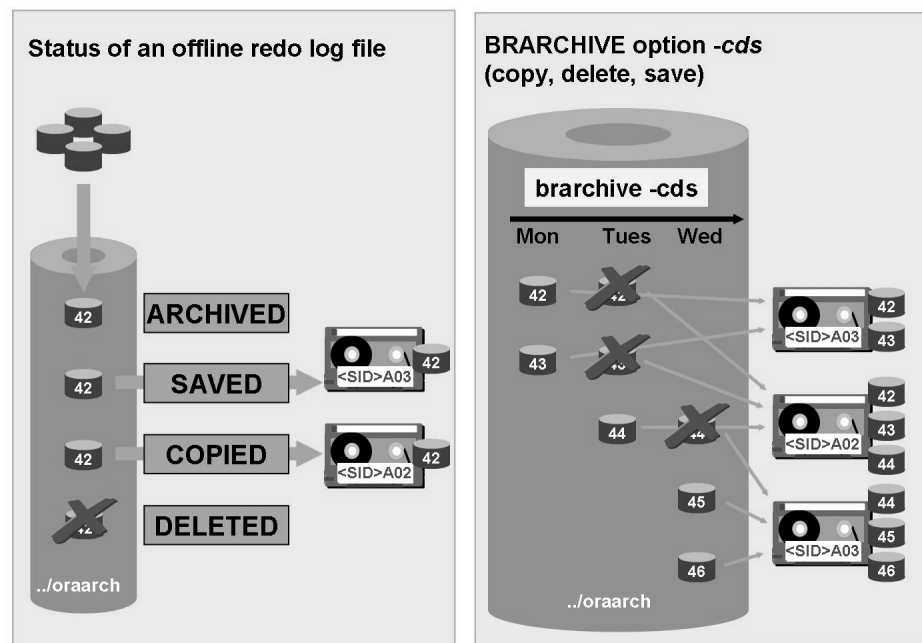


Figure 78: Backup and Status of Offline Redo Log Files

An offline redo log file can have various statuses for BRARCHIVE. These statuses are always updated in the summary log `arch<DBSID>.log` after a BRARCHIVE run.

During a backup to tape, an offline redo log file has the status ARCHIVE. At first save, the file status is SAVED; the second time, it is COPIED; and after deletion it has the status DELETED.

During a backup to disk, an offline redo log file has the status DISK. A second copy is not supported. The only statuses here are DISKSAV (first save to disk) and DISKDEL (deletion after a save to disk).

BRARCHIVE has several call options (functions) that determine how the offline redo log files are processed. SAP recommends using the option `-c ds` (copy_delete_save), which is also the default option when starting BRARCHIVE from the DBA Cockpit or transaction DB13.

First, all offline redo log files with status SAVED are saved to tape for a second time, and subsequently deleted from disk. Then, all offline redo log files with status ARCHIVE are backed up to tape for the first time, and their status is changed to SAVED.

After the backup, all offline redo log files exist at two locations: either in directory `oraarch` and on tape, or on two different tapes. Thus, you can achieve a high safety rate without drastically increasing the tape requirement.

To perform backups of the archived redo logs, start BRTOOLS or BRGUI and choose *Backup and database copy* → *Archivelog backup*. The following menu is shown to select options and parameters; most of them are similar to BRBACKUP:.



Performing Backups of the Archived Redo Logs Using BR*Tools

```
BR0657I Input menu 15 - please check/enter input values
```

```
-----
BRBACKUP main options for backup and database copy
```

```
1 - BRBACKUP profile (profile) ..... [initT99.sap]
2 - Backup device type (device) ..... [disk]
3 # Tape volumes for backup (volume) . []
4 # BACKINT/Mount profile (parfile) .. []
5 - Database user/password (user) .... [/]
6 - Backup type (type) ..... [online]
7 # Disk backup for backup (backup) .. [no]
8 # Delete disk backup (delete) ..... [no]
9 ~ Files for backup (mode) ..... [all]
```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

Select the function of BRARCHIVE from the *BRARCHIVE function (function)* menu. Nearly any combination of save, copy, and delete is possible. Using different functions, you can, for example:

- Implement a two-phase strategy: the first run saves new archived redo logs (save), and the second run creates a second backup and deletes archives successfully backed up twice (second_copy_delete).
- Create a parallel backup on two different tape stations (double_save) and later delete (delete_copied).



Caution: No matter which strategy is used, for security reasons at least two copies of archived redo logs should exist at any time.

Verifying Backups of Offline Redo Log Files

Database archive log files and their backups can be verified with BR*Tools 7.00 as of patch 22 with the Oracle Recovery Manager (RMAN). See also SAP Note 1016173. The RMAN VALIDATE command is called internally. This is especially important for archive log files because until now there has been no way of verifying internal consistency. RMAN verification on the other hand covers database files with the DBVERIFY function, which means that RMAN does not offer any essential advantages compared to DBVERIFY.

RMAN verifications can be activated using the following command options:

BRARCHIVE: -w | -verify use_rmv | first_rmv | only_rmv

The option first_rmv verifies the original files with RMAN before the archive log files are backed up.

Verifications of archive log files with RMAN will be supported as of Basis 7.00 Support Package 12 in the DBA Cockpit.



Hint: This new functionality can also be used for Oracle 9i after BR*Tools 7.00 has been correctly installed in this environment (see SAP Note 849483).

BRBACKUP and BRARCHIVE: One-Run Strategy

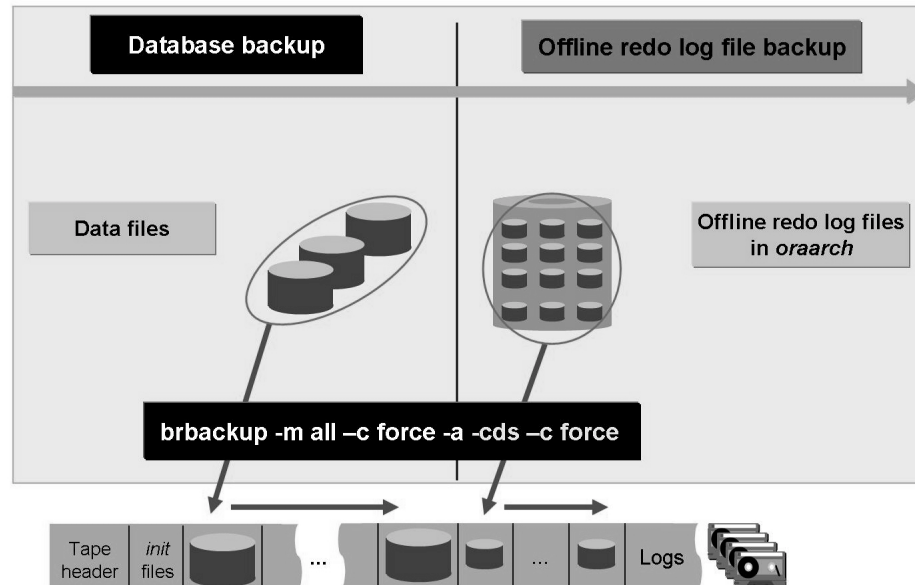


Figure 79: BRBACKUP and BRARCHIVE: One-Run Strategy

The advantage of the one-run strategy is that you can create a complete database backup and an offline redo log backup in one backup procedure. BRBACKUP and BRARCHIVE are called together rather than individually. Only one tape pool (in this case the one defined in parameter `volume_backup`) is used. The offline redo log files are backed up to the tapes where the database files are backed up. This saves tapes and reduces management costs.

To define the one-run strategy for BRBACKUP, use the option `-a | -archive`. After this, the options for BRARCHIVE follow. A corresponding job can be defined in the DBA Cockpit or transaction DB13.

With this procedure, BRBACKUP backs up all database files as usually, and then it starts BRARCHIVE, passing to it the options entered after `-a | -archive`. BRARCHIVE first backs up the corresponding offline redo log files (as usual) and then it backs up all logs, including BRBACKUP logs.

With the one-run strategy, the maximum number of offline redo log files that can be backed up is the number that can still fit on the BRBACKUP tape after the database backup. If more offline redo log files are generated daily than can be backed up, for example because the database has grown or the number of offline redo log files is increasing, the archiver gets stuck (and therefore, the database). This situation

is called “archiver stuck”. Therefore, you must regularly check whether the tape capacity is sufficient. If necessary, you should use larger tapes, an extra tape station, or another backup strategy.



Caution: The one-run strategy cannot be used to resolve an archiver stuck since BRBACKUP attempts to connect to the database. If an archiver stuck is to be resolved using BRARCHIVE, tapes must be available in tape pool `volume_archive`.

Consistent Online Backups

A consistent online backup is a database backup in online mode that contains logically consistent data. In this case, the offline redo log files generated during the backup are saved to the same volume as the database files that are backed up with BRBACKUP.

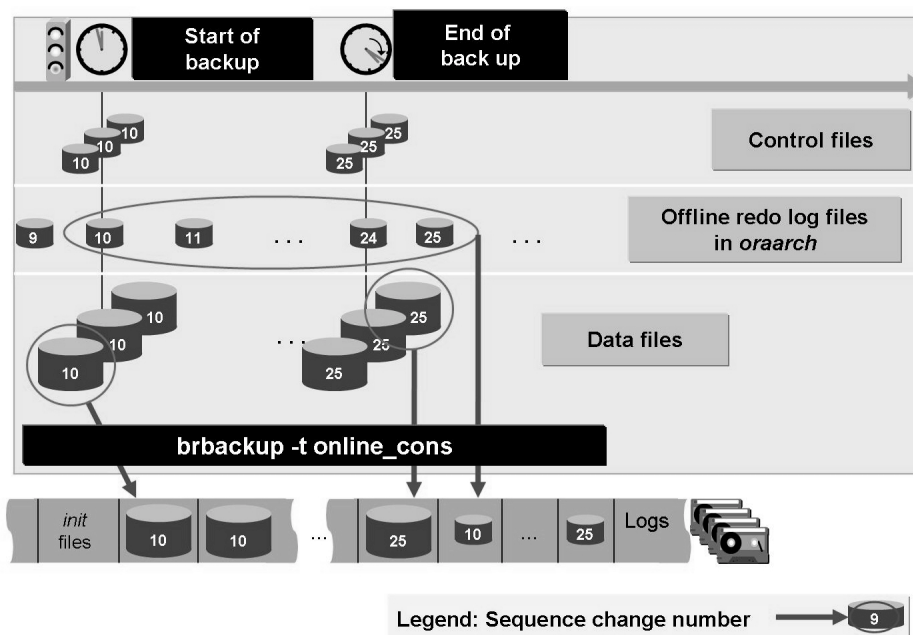


Figure 80: Consistent Online Backups

After backing up all data files online, BRBACKUP performs a log switch. BRBACKUP waits until the archiver process has finished copying the last redo log file into directory `oraarch`, and then copies the offline redo log files created during the online backup to tape. The last files on tape are the BRBACKUP summary and detail log.

The backup of the offline redo log files in a consistent online backup is completely controlled by BRBACKUP. Therefore, this run is independent of the BRARCHIVE backups, and does not affect them. In particular, no entries are created in the arch<DBSID>.log summary log.

A consistent online backup can be performed either as a whole backup, a full backup, or an incremental backup. This cannot be scheduled in the DBA Cockpit.

A consistent online backup can be used to reset the database to its status at the end of the backup. This is done by restoring data files and offline redo log files, and performing a point-in-time recovery.



Hint: A consistent online backup is usually used for special backups that are done once a month or per quarter and put in long-term storage. It is also recommended to perform this backup before an SAP database upgrade.

Checking Backup Logs

You should regularly check the result of all backups.

- The log viewer for DBA operations (transaction DB14) is the main tool to check these results, as this is a single place to view logs of all DBA activities.
- You can also use the DBA planning calendar in the DBA Cockpit (DB13). In the calendar, you see the scheduled actions, with colors if they have warnings or errors.
- To view only backup logs, use transaction DB12. This transaction can also be used to create a recovery report, to view the status of the archiving directory, and to get an overview over archived redo log files.

Scheduling Backups from the DBA Planning Calendar

For regular backups (and other regular database actions) use the DBA Planning Calendar (transaction DB13) in the DBA Cockpit. From the action templates, you can schedule any useful combination of:

- Whole Backups
- Online Backups
- Offline Backups
- Partial Backups
- Full Backups
- Incremental Backups
- Redo Log Backups

All templates offering a backup plus a redo log backup will perform BRBACKUP and BRARCHIVE in one run.

Actions for which backups are planned such as tape initialization, determining compression rates, or the preparation run for RMAN backup can also be scheduled with transaction DB13.



Hint: To use the DBA Planning Calendar, check that parameters in `init<DBSID>.sap` are maintained correctly. While parameters such as tape names can be specified when planning a backup with DB13, parameters like `device_type` or `tape_adress` cannot be selected from DB13.

Exercise 7: Performing Backups

Exercise Objectives

After completing this exercise, you will be able to:

- Perform various types of backups

Business Example

You want to learn how to perform different types of backup.

Task:

Perform various types of backups.

1. Perform a complete offline backup of the database.
2. Perform a complete online backup of the database.
3. Perform a backup of offline redo log files. Since the data is backed up on disk, save the offline redo log files, and delete them.

Solution 7: Performing Backups

Task:

Perform various types of backups.

1. Perform a complete offline backup of the database.
 - a) Start BRGUI or BRTOOLS and choose *Backup and database copy* → *Database backup*.
 - b) In the input menu BRBACKUP main options for backup and database copy, select backup type offline and backup mode all.
2. Perform a complete online backup of the database.
 - a) Start BRGUI or BRTOOLS and choose *Backup and database copy* → *Database backup*.
 - b) In the input menu BRBACKUP main options for backup and database copy, select backup type online and backup mode all.
3. Perform a backup of offline redo log files. Since the data is backed up on disk, save the offline redo log files, and delete them.
 - a) Start BRGUI or BRTOOLS and choose *Backup and database copy* → *Archivelog backup*.
 - b) In the input menu BRARCHIVE main options for archivelog backup, select the BRARCHIVE function save_delete.



Lesson Summary

You should now be able to:

- Perform online, offline, and partial backups
- Perform RMAN backups, including incremental backups
- Create backups of archived redo log files

Lesson: Restore and Recovery

Lesson Overview

This lesson introduces several restore and recovery scenarios and explains how to restore and recover a database using BR*Tools.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe a potential problem that would lead to a restore/recovery scenario
- Perform a complete recovery of the database
- Perform a point-in-time recovery of the database
- Perform a disaster restore/recovery of the database

Business Example

Due to a disk crash on a non-mirrored disk containing data files, the database cannot be started anymore. After replacing the disk, you must perform a restore of the missing files and recover the database.

Introduction

In an SAP system with an Oracle database, all data files have the status online and read/write. For a functioning, consistent database, all data files and the control file must be synchronized, that is, their times must match.

In Oracle, files are synchronized using timestamps. Timestamps are integers that are increased during certain database actions, and entered in all data and control file headers by the log writer or checkpoint process at the checkpoint event.

An example of synchronization data is the log sequence number (LSN), which is increased by 1 during every log switch. At a more sophisticated level, Oracle defines synchronization on transaction level using the system change number (SCN), which is increased, for example, after the COMMIT in a modifying transaction, or at the checkpoint.

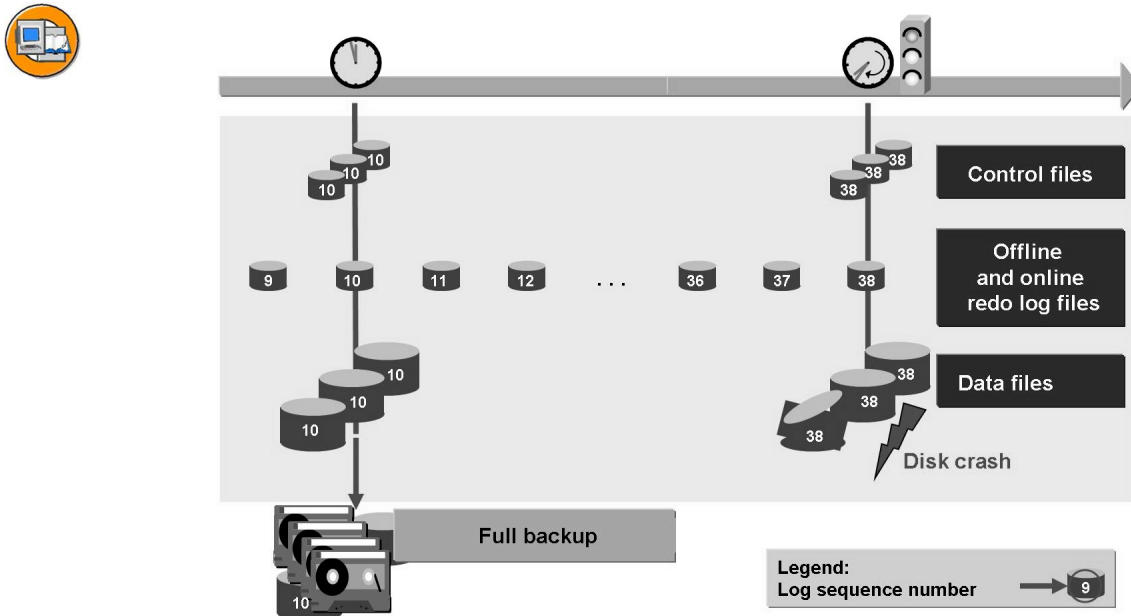


Figure 81: Introduction

The figure above shows an example of a database that was fully saved without errors at the time point LSN=10. At the time point LSN=38, the database was destroyed by a media or user error in such a way that the database instance fails or the database becomes inconsistent. The offline and online redo log files that were created between the beginning of the backup and the occurrence of the error are available, and they are indispensable for recreating the data in the database.

**Have a problem-solving strategy!****Do not make any rash decisions.****Analyze the problem in detail.****Before restoring any files, check:**

- What is causing the problem
- Whether there is enough disk space to save and restore files
- Whether a hardware extension is necessary
- The file system and mount points
- The availability of backups
- The availability of offline redo log files

Figure 82: Problem Handling

If a database problem occurs, you must analyze the problem and create a problem-solving strategy. For typical problem situations, you should have escalation plans ready and tested.

To analyze the database problem, check the database alert log and trace files belonging to the background processes in directory `$ORACLE_HOME/saptrace/background`.

Your problem-solving strategy depends on the answers to the following questions:

- What is the status of the database – available or not available?
- Is this a user error or a media error?
- Which files were destroyed?
- Which file types (data files, control files, online redo log files) are affected?
- Is software or hardware mirroring available?

Using backup strategies recommended by SAP, you will have a number of database backups and offline redo log file backups for a restore and recovery. Your problem-solving strategy will determine which backup and offline redo log files are copied back, and how they need to be applied.

To be on the safe side (and time permitting), perform a complete offline backup before the files are copied back in the restore phase using BRBACKUP (if the database is running properly) or operating system backup tools. This is especially important when you are going to perform a point-in-time recovery or a database reset because these strategies always involve data loss. In addition, all offline redo log files in oraarch should be saved with BRARCHIVE (but not deleted).

In the event of a hard disk problem, such as a head crash, you must:

- Replace hardware
- Create volume(s) on hard disks
- Create file systems and mount them at the old locations



Caution: Do not make any rash decisions. If you make mistakes or act carelessly, you can drastically aggravate the restore and recovery situation. The costs incurred by a consulting session provided by SAP or an SAP partner are negligible compared to the business consequences of data loss, even for a single day of production operation.



Transaction DB12

Backup Logs: Overview for Database DEV

Refresh Recovery report

Backup Logs

DB Name	DEV	Started	15.02.2008
DB Server	TWDF1902		12:55:55
DB Release	10.2.0.2.0		

Database backups

Last successful backup 21.01.2008 09:13:26

Overview of database backups

Redo log backups

Archiving directory status

Overview of redo log files Not yet backed up: 53

Overview of redo log backups

Figure 83: Recovery Report

To test the reliability of your backup strategy, run the Recovery report in SAP transaction DB12 on a regular basis. It provides important information that can be used in the event of an Oracle database failure requiring database recovery.

When you start the report, the system displays information about the last successful backup, including backup type and tape names. This tells you which backup to use for a recovery. The report also checks whether the required redo log files are available (backed up on tape or in the archiving directory). Therefore, you know which files must be restored in the event of a recovery.

Checking the recovery report regularly helps you to detect possible gaps in your backups:

- Missing redo log files are very dangerous because if an error occurs, the database can no longer be restored to the current point in time. You must perform a complete database backup as soon as possible to resolve this critical situation.
- If the list of redo log files is too long, a recovery to the current status may take a long time. In this case, you should likewise perform a full database backup as soon as possible.

Concepts: Complete Database Recovery

Typical problem scenario: Because of a head crash, data has been lost during business operation. The database is inconsistent, and it is no longer running properly.

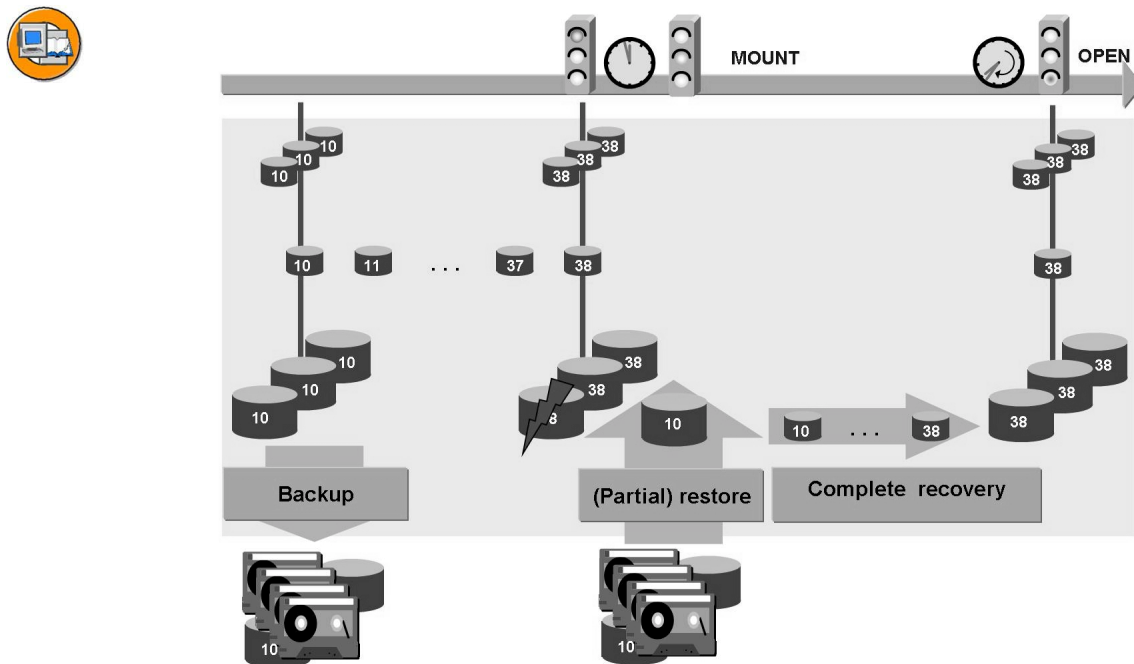


Figure 84: Concepts: Full Database Recovery

A complete database recovery is performed to restore missing data files and to recover the database to its (committed) status just before the error occurred.

During a restore, database files are copied from the backup medium back to the disk. Using the complete database recovery strategy, only the required minimum of data is copied. The database files that are to be copied back can be combined from different backups. Because the database files are no longer synchronous after a partial restore, the database is inconsistent and cannot run properly after the copy-back procedure has terminated.

To synchronize the files, the database evaluates the synchronization data that has been saved in the file headers. The database requests all offline redo log files that have accumulated since the **oldest** database file (in logical terms), in uninterrupted sequence. During a recovery, all data changes logged by these offline redo log files are “replicated” in the files that have been copied back from a backup medium.

With complete database recovery, all changes are performed again until all data files are at the same SCN. This procedure is called **media recovery**. When the database is subsequently started up, during the instance recovery all open transactions are first rolled forward, and those that are not committed in the redo log are taken (rolled) back using the undo space (which is likewise recovered). After the instance recovery, the database is consistent, capable of running, and is back to its committed data status. (period missing)

Concepts: Point-in-Time Recovery

Typical problem scenario: During an upgrade, a user accidentally drops a table. As a result, the upgrade must be terminated. A complete backup is available, but was it not created immediately before the upgrade process began.

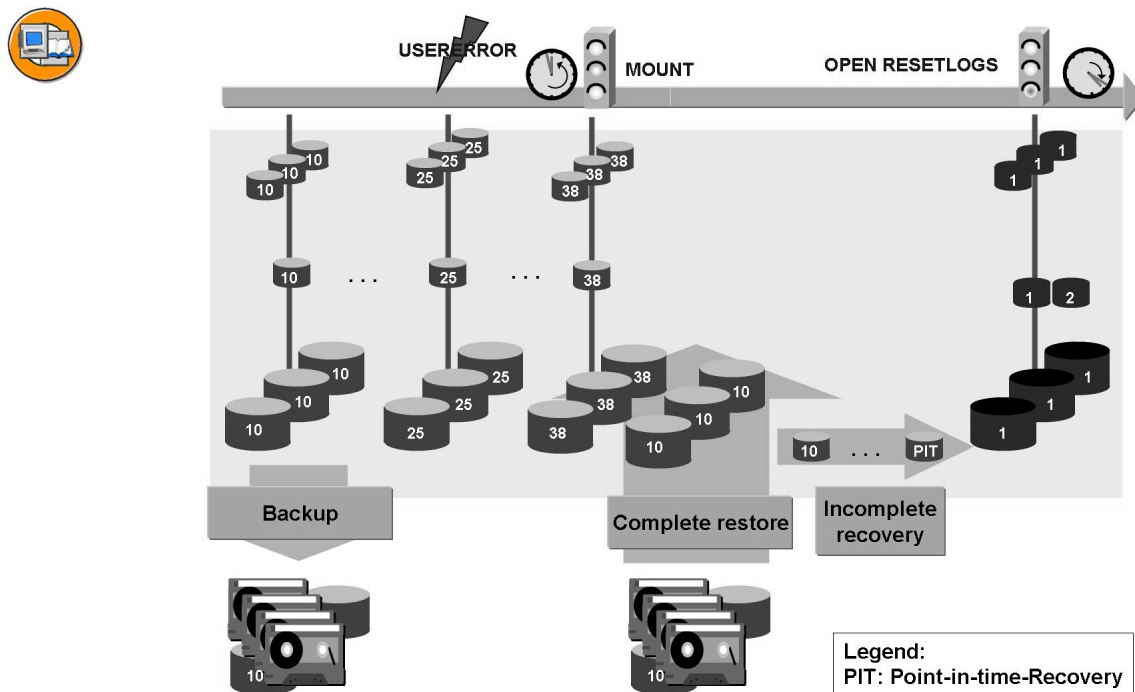


Figure 85: Concepts: Database Point-in-Time Recovery

A point-in-time recovery is performed to reset the database to the status at a certain point in time before the upgrade with help of a complete backup, and then to recover the data up to a later, appropriate point in time (for example, up to the start of the upgrade, or up to the table drop).

Initially, all data files are replaced by copies from a complete online/offline backup (or from a group of partial backups that cover the whole database). The termination point of the recovery determines whether the control files should also be replaced. The names of all data files and online log files including their corresponding paths are in the control file. The file names in the control files must match the file structure after the recovery has finished on the operating system level.

During the recovery phase, the changes to the database are performed again. Incomplete recovery refers to the end point of recovery, which can be anywhere between the end of the copied backup and the last entry in the current online redo log. The recovery end point can be defined by the redo log file sequence number (LSN), by the sequence change number (SCN), or by specifying a point in time.

A point-in-time recovery always results in data loss. The data that was generated between the chosen point in time and the time of last shutdown is lost.



Caution: After a point in time recovery, unless a complete recovery is performed, the database is usually opened using the Oracle command ALTER DATABASE OPEN RESETLOGS (called by BRRECOVER), which resets the online redo log files and the LSN to an initial status. Therefore, the old redo logs and the new ones do not form a sequence of logs that could be used for a complete recovery based on the same old backup. A complete backup must be triggered immediately in a production database before you start using it.

A point-in-time recovery can be performed for the whole database or just for a set of tablespaces. For databases not having Multiple Components in One Database (MCOD), a database point-in-time recovery usually has to be performed. Tablespace recovery allows you to restore a tablespace for an individual component in MCODE databases without damaging the remaining components in the database.

Concepts: Whole Database Reset

Typical problem scenario: During an upgrade, extensive software or hardware problems arise. As a result, the upgrade must be terminated. The database is inconsistent, and it is no longer running properly. A complete (offline or consistent online) backup is available, created immediately before the upgrade process began.

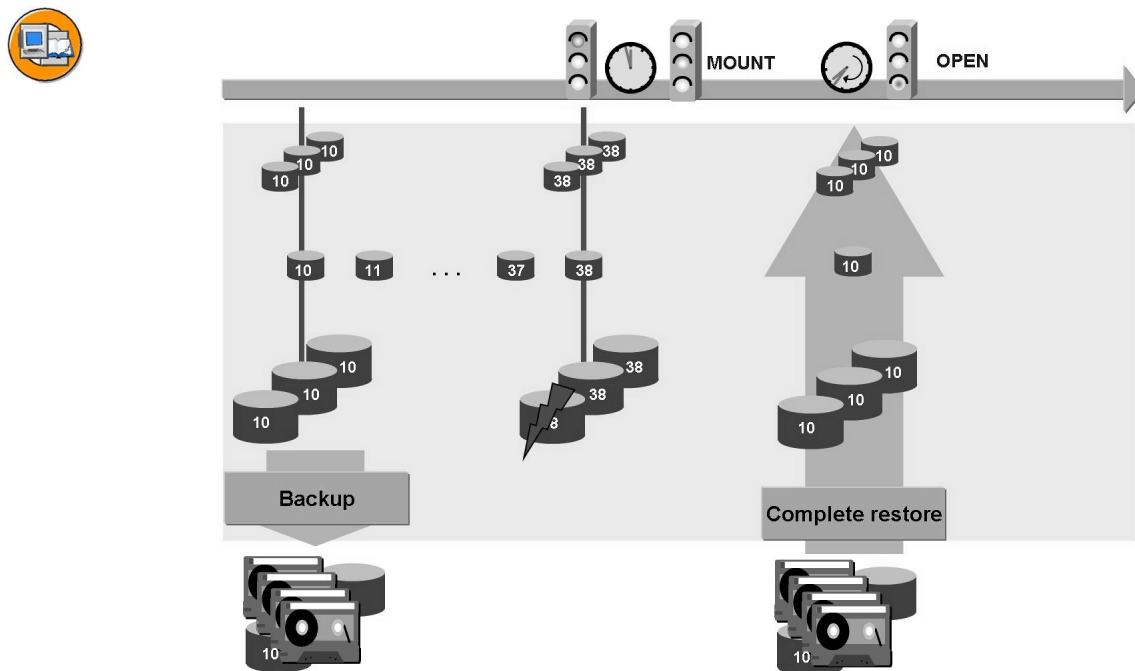


Figure 86: Concepts: Whole Database Reset

A whole database reset is performed to reset the database to its status corresponding to the end of the complete backup, that is, to the status immediately before the upgrade.

When the database is reset, all data files, online redo log files, and control files are copied from the backup medium. If all these files come from the same valid offline backup, the database is consistent and ready for operation after the copy process has finished. A recovery is not required, and the database can be started immediately. If you reset from a consistent online backup, recovery is automatically performed up to the end point of the backup.

Like point-in-time recovery, database reset always results in data loss. The data that has been generated after the applied complete backup is lost. Of course, the database as such does remain consistent.

Complete Database Recovery with BR*Tools

When a complete database recovery is performed, BRRECOVER replaces lost data files by using appropriate backups, and subsequently recovers the restored data file status using redo log files. To be able to use this function, your online redo log files and control files must be valid.

➔ **Note:** Complete database recovery with BRRECOVER is considered a safe procedure, which means you cannot cause more damage to the database by performing this procedure than you already have. Therefore, you can use this procedure for other problem scenarios as well, even for scenarios which do not require a restore but only a recovery (like a database crash during an online backup or when a tablespace went offline). BRRECOVER will never restore any file if not necessary.

To perform a complete database recovery, start BRTOOLS or BRGUI and choose *Restore and recovery* → *Complete database recovery*. In the next menus provided by BRTOOLS, you can enter parameters for the recovery. If you enter nothing here (which is the normal procedure), all required input can be done in list and selection menus provided by BRRECOVER later.

The complete database recovery procedure consists of several phases. BRRECOVER presents them in the main menu of complete database recovery, and they must be executed in the predetermined sequence – that is, a particular phase can only be selected after the previous one has been successfully completed.



Complete Database Recovery Using BR*Tools

```
BR0655I Control menu 101 - please decide how to proceed
```

```
-----
Complete database recovery main menu
```

- 1 = Check the status of database files
- 2 * Select database backup
- 3 * Restore data files
- 4 * Restore and apply incremental backup
- 5 * Restore and apply archivelog files
- 6 * Open database and post-processing
- 7 * Exit program
- 8 - Reset program status

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
```

```
-----
BR0662I Enter your choice:
```

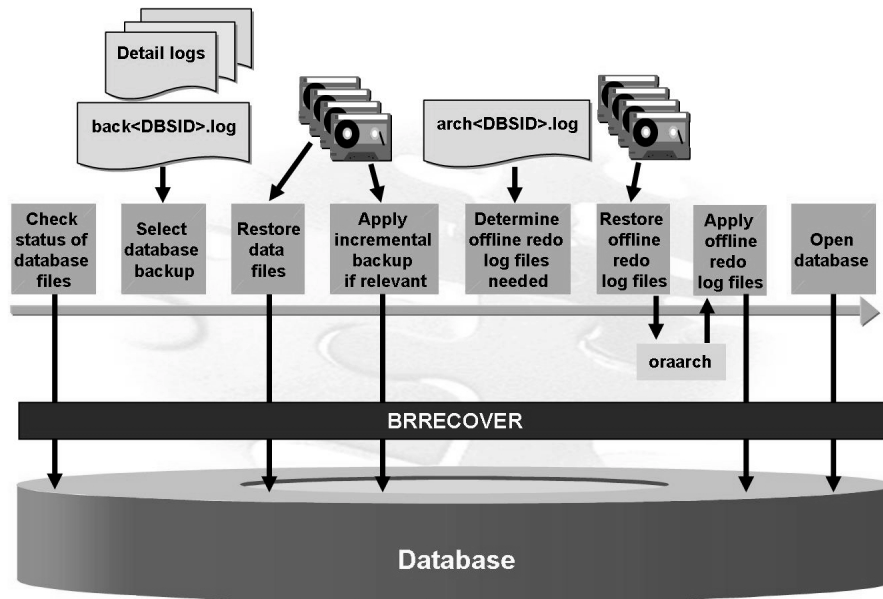


Figure 87: Complete Database Recovery with BR*Tools

Check the status of database files

BRRECOVER checks the status of all files in the database (that is, the control files, online redo log files, and data files). BRRECOVER does the following:

- To update the Oracle's dynamic V\$ views, which are reloaded during startup to NOMOUNT and MOUNT, BRRECOVER stops the database instance if it is started and starts it to MOUNT status.
- BRRECOVER refers to entries in some V\$ views, such as V\$DATAFILE and V\$RECOVER_FILE, to determine the status of database files.
- BRRECOVER logs any errors concerning data files to the detail log created in the sapbackup directory. This log gets the suffix (function ID) crv for complete recovery.

Select database backup

BRRECOVER determines the eligible backups using the entries in the BRBACKUP summary log file `back<DBSID>.log` (eligible backups are those with return code 0 or 1). The associated detail logs show whether the data files required for the restore of missing files are in the backup.

Missing data files can be restored from various backups during one recovery process. To minimize the subsequent recovery time, BRRECOVER always suggests the most recent backup.

You can also select an incremental backup to be restored before applying offline redo log files. In this case, BRRECOVER automatically selects the corresponding full backup to restore missing files.

BRRECOVER also roughly checks the availability of offline redo log files.

Restore data files

BRRECOVER calls BRRESTORE to restore the data files to their original location.



Caution: Neither BRRECOVER nor BRRESTORE creates missing `sapdata` directories automatically, so you must create them manually on operating system level before you start restoring missing files. However, BRRESTORE automatically creates missing `sapdata` subdirectories during this phase.

Restore and apply incremental backup

If you selected an incremental backup during the Select database backups phase, BRRECOVER calls BRRESTORE to restore and apply the selected incremental backup.

Restore and apply archivelog files

BRRECOVER determines the offline redo log files required for a complete recovery. The BRARCHIVE summary log file `arch<DBSID>.log` lists the backups of the offline redo log files.

BRRECOVER takes into consideration existing offline redo log files in `oraarch` (or `saparch` in older releases), as well as online redo log files.

BRRECOVER then calls BRRESTORE to restore the offline redo log files that have been found in backups back to the `oraarch` (or `saparch`) directory.

Finally, BRRECOVER calls SQL*Plus to apply offline redo log files to the database (Oracle statement `RECOVER DATABASE`).

- Offline redo log files are applied to the database in groups of at most 100 files. If you have more than 100 files to apply, the restore and apply phase is repeated automatically as necessary.
- The restore and apply phases can be executed in parallel to minimize total recovery time.

BRRECOVER is able to reprocess a structural change in the database, such as an extension of a tablespace by a new file. Contrary to older releases, it is not necessary to create a new backup after a structural change.

Open database

During the last phase, BRRECOVER opens the database and checks the status of database files and tablespaces.

Point-in-Time Recovery

To perform a point-in-time recovery, start BRTOOLS or BRGUI and choose *Restore and recovery* → *Database point-in-time recovery*. In the next menus provided by BRTOOLS you can enter parameters for the recovery. If you enter nothing here (which is the normal procedure), all required input can be done in list and selection menus provided by BRRECOVER later.

The database point-in-time recovery procedure consists of several phases. BRRECOVER presents them in the main menu of point-in-time recovery, and they must be executed in the predetermined sequence – that is, a particular phase can only be selected after the previous one has been completed successfully:

```
BR0655I Control menu 103 - please decide how to proceed
```

```
-----
Database point-in-time recovery main menu
```

- ```
1 = Set point-in-time for recovery
2 * Select database backup
```

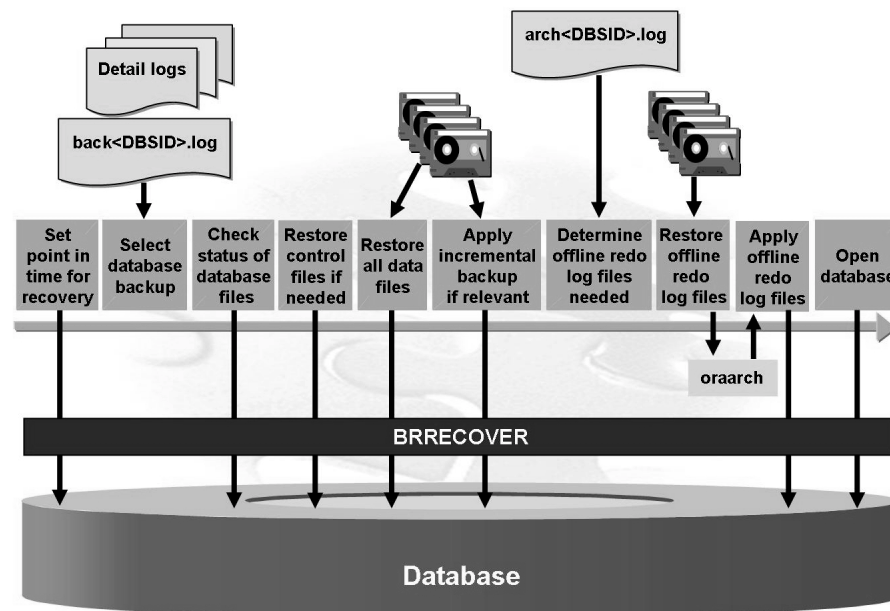
```

3 * Check the status of database files
4 * Restore control files
5 * Restore data files
6 * Restore split control files
7 * Restore and apply incremental backup
8 * Restore and apply archivelog files
9 * Open database and post-processing
10 * Exit program
11 - Reset program status

```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

-----  
BR0662I Enter your choice:



**Figure 88: Database Point-in-Time Recovery with BR\*Tools**

### Set point-in-time for recovery

BRRECOVER lets you enter the recovery end point by choosing one of the following:

- Redo log sequence number (LSN)
- System change number (SCN)
- Point in time

**Select database backup**

BRRECOVER determines the eligible backups using the entries in the BRBACKUP summary log file `back<DBSID>.log` (return code 0 or 1). The associated detail logs show which data files were saved in which backup.

During database point-in-time recovery, a complete restore must be carried out, so all data files are needed. They can be restored from different backups. To minimize the subsequent recovery time, BRRECOVER always suggests the most recent complete backup.

BRRECOVER also roughly checks the availability of offline redo log files.

You can also select an incremental backup to be restored before applying offline redo log files. In this case, BRRECOVER automatically selects the corresponding full backup to restore all data files.

**Check the status of database files**

BRRECOVER checks the status of all files in the database (control files, online redo log files, and data files) to determine which ones will be overwritten and which ones recreated. To update the V\$ views, BRRECOVER stops the database instance if it is started and starts it to MOUNT status if control files are available.

**Restore control files**

BRRECOVER calls BRRESTORE to restore control files if needed, that is, if they are unavailable or unsuitable for the selected backups.

**Restore data files**

BRRECOVER calls BRRESTORE to restore the data files to their original location.

**Restore and apply incremental backup**

If you selected an incremental backup during the Select database backups phase, BRRECOVER calls BRRESTORE to restore and apply the selected incremental backup.

**Restore and apply archive log files**

BRRECOVER determines the offline redo log files required for the recovery up to the indicated time point. The BRARCHIVE summary log file `arch<DBSID>.log` lists the backups of the offline redo log files. BRRECOVER takes into consideration existing offline redo log files in `oraarch` (or `saparch`), as well as online redo log files.

BRRECOVER then calls BRRESTORE to restore the offline redo log files that were found in backups back to the `oraarch` (or `saparch`) directory.

Finally, BRRECOVER calls SQL\*Plus to apply redo log files to the database (Oracle statement `RECOVER DATABASE UNTIL ...`).

### Open database

During the last phase, BRRECOVER:

- Opens the database with the option RESETLOGS (this is required because of incomplete recovery)
- Creates missing temporary files
- Checks the status of database files and tablespaces
- Deletes unnecessary files that are no longer used by the database

### Whole Database Reset

To perform a complete database recovery, start BRTOOLS or BRGUI and choose *Restore and recovery* → *Whole database reset*. In the next menus provided by BRTOOLS you can enter parameters for the database reset. If you enter nothing here (which is the normal procedure), all required input can be done in list and selection menus provided by BRRECOVER later.

The whole database reset procedure consists of several phases. BRRECOVER presents them in the main menu, and they must be executed in the predetermined sequence – that is, a particular phase can only be selected after the previous one has been successfully completed.

```
BR0655I Control menu 109 - please decide how to proceed
```

```

Whole database reset main menu
```

- ```
1 = Select consistent database backup  
2 * Check the status of database files  
3 * Restore control files and redolog files  
4 * Restore data files  
5 * Restore and apply incremental backup  
6 * Apply archive log files  
7 * Open database and post-processing  
8 * Exit program  
9 - Reset program status
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
```

```
-----  
BR0662I Enter your choice:
```

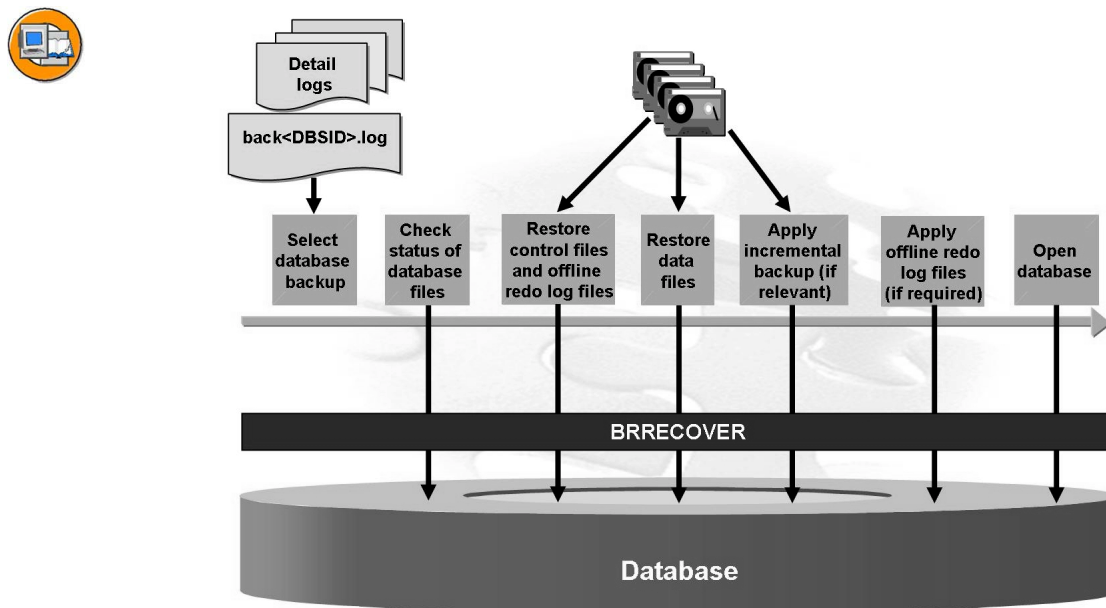


Figure 89: Whole Database Reset with BR*Tools

A BRRECOVER whole database reset goes through the following steps:

Select consistent database backup

BRRECOVER determines the most suitable backups using entries in the BRBACKUP summary logback<DBSID>.log (return code 0 or 1) in BRBACKUP. You can select:

- A complete offline backup
- A complete consistent online backup
- An incremental offline backup
- An incremental consistent online backup

If you choose an incremental backup, BRRECOVER automatically selects the corresponding full backup to restore all data files.

Restore control files and redolog files

BRRECOVER calls BRRESTORE to restore control files. Offline redo log files are also restored if a consistent online backup was selected.

Restore data files

BRRECOVER calls BRRESTORE to restore the data files to their original location.

Apply incremental backup

If you selected an incremental backup during the Select consistent database backup phase, BRRECOVER calls BRRESTORE to restore and apply the selected incremental backup.

Apply archivelog files

If a consistent online backup was selected, BRRECOVER calls SQL*Plus to apply the restored offline redo log files to the database.

Open database

During this phase, BRRECOVER:

- Opens the database (if necessary, with the option RESETLOGS)
- Creates missing temporary files
- Checks the status of database files and tablespaces
- Deletes unnecessary files that are no longer used by the database

Disaster Recovery

If you lose your entire Oracle database system (possibly including hardware) and have not taken any special security precautions, such as setting up an Oracle standby database, then you have to recover the whole system, step by step. Since the restore and recovery procedures described above all depend on the existence of configuration profiles and backup log files, there is a special procedure called disaster recovery embedded in BRRECOVER that is able to restore these files when they are missing.



Note: Disaster recovery is only a preparation step for a subsequent database recovery with database point-in-time recovery or whole database reset.

Prerequisites for using disaster recovery are:

- SAP and Oracle software is correctly installed.
- File systems with the sapdata directories exist and are configured as before the “disaster”.

To perform a disaster database recovery, start BRTOOLS or BRGUI and choose *Restore and recovery* → *Disaster recovery*. In the next menus provided by BRTOOLS, you can already insert parameters for the disaster recovery. If you enter nothing here (which is the normal procedure), all required input can be done in list and selection menus provided by BRRECOVER later.

```
BR0656I Choice menu 136 - please make a selection
```

```
-----
Disaster recovery main menu
```

- 1 = Restore profiles and log files from BRBACKUP backup
- 2 - Restore profiles and log files from BRARCHIVE backup
- 3 * Exit program
- 4 - Reset program status

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

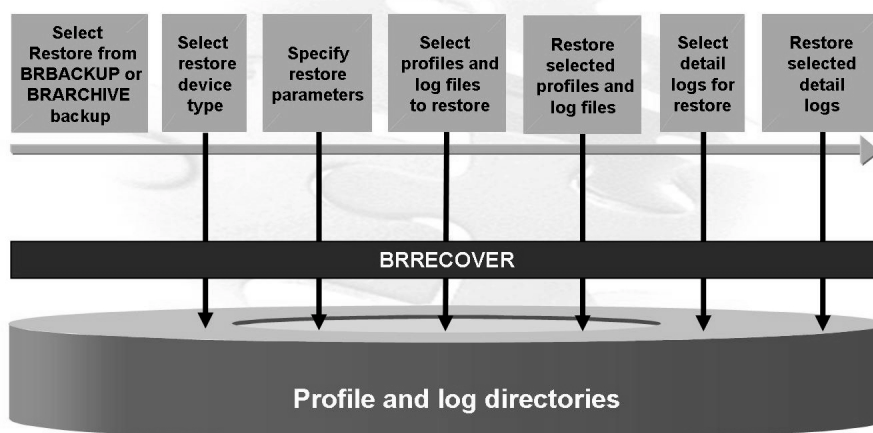


Figure 90: Disaster Recovery with BR*Tools

To perform a disaster recovery, BRRECOVER first asks from where the files are to be restored. In other recovery scenarios, this information is read from `init<DBSID>.sap` and the logs. In this case, these files are gone and are restored in the first step.

After restoring `init<DBSID>.sap`, the backup summary log `back<DBSID>.log` is restored so that the system knows from which backups the other files can be restored.

The other profiles and log files to be restored can now be selected from a list. In normal cases there is no need to change the default selection, as BRRECOVER has already determined which files need to be restored and which are not needed.

In the next selection, Restore of BRBACKUP detail logs, a list of backup detail logs is displayed to be selected for restore. You should only select the detail logs of the backup or backups you need to restore the database later, because for every selected backup, the corresponding backup tape must be mounted to restore the detail log.

Other Functions of BRRECOVER

When calling BRTOOLS or BRGUI and choosing *Restore and recovery*, two functions are displayed which were not yet covered in this lesson.



Caution: Use these functions very carefully and only if you fully understand the procedure. These functions are "expert functions", which should only be used in exceptional cases.

Restore of individual backup files

Use this function to restore individual files from a backup, for example, to perform a manual restore and a manual recovery using the function *Restore and application of archive log files*.

Restore and application of archive log files

Use this function to perform a manual recovery, for example, after performing a manual restore using a *Restore of individual backup files*.



Hint: For any other scenarios which cannot be resolved by BRRECOVER, use the procedures described in *SAP Library - SAP Database Guide: Oracle*.

Exercise 8: Restore and Recovery

Exercise Objectives

After completing this exercise, you will be able to:

- Perform a complete database recovery
- Perform a point-in-time recovery
- Perform a database reset

Business Example

A disk has become unusable. You want to restore and recover the database to its most recent state.

Task:

In this exercise, various restore and recovery scenarios are performed. Before deleting any files to perform the exercises, make sure that you have proper offline and online backup, and that the database is running in archive log mode!

Because Windows does not allow you to delete a file in use, you must shut down Oracle before deleting a file to simulate a disk crash.

1. Simulate a disk crash by shutting down your database and deleting data file `G:\oracle\<DBSID>\sapdata3\<DBSID>_1\<DBSID>.data1`. Then start the database and check the error message. Decide which scenario you want to use to recover the database and perform the recovery.
2. To simulate a user error, start the script `usererror.bat`, located in `G:\oracle\<DBSID>\scripts`. This script will display the current system time, which you should note. After waiting 60 seconds, the script will first display the number of rows of table *DBCHECKORA*, then drop *DBCHECKORA*. To show, that *DBCHECKORA* was dropped, the command to show the number of rows in *DBCHECKORA* is executed again, but will show an error message indicating that *DBCHECKORA* does not exist.

Which is the correct scenario to recover the database? Perform the recovery.

3. **(Optional)** Repeat the last exercise. Recovery will not be possible until the point-in-time of the user error. Why?
4. Because you might have messed up the database, you decide to reset the database to the point in time of the offline backup taken in the last exercise. Perform the correct scenario.

Solution 8: Restore and Recovery

Task:

In this exercise, various restore and recovery scenarios are performed. Before deleting any files to perform the exercises, make sure that you have proper offline and online backup, and that the database is running in archive log mode!

Because Windows does not allow you to delete a file in use, you must shut down Oracle before deleting a file to simulate a disk crash.

1. Simulate a disk crash by shutting down your database and deleting data file
G:\oracle\<DBSID>\sapdata3\<DBSID>_1\<DBSID>.data1.
Then start the database and check the error message. Decide which scenario you want to use to recover the database and perform the recovery.

- a) Shut down the database using, for example, `brspace -c force -f dbshut`.

Delete the data file

```
G:\oracle\<DBSID>del sapdata3\<DBSID>_1\<DBSID>.data1
```

- b) Start the database using, for example, `brspace -c force -f dbstart` and check the error message:

```
BR0613I Database instance T99 is shut down
```

```
BR0786I Database instance T99 will be opened now in mode 'normal'
```

```
BR0280I BRSPACE time stamp: 2007-11-26 15.52.17
```

```
BR0304I Starting and opening database instance T99 ...
```

```
BR0278E Command output of 'g:\oracle\DEV\102\BIN\sqlplus':
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 15:52:17 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
SQL> Connected to an idle instance.
```

```
SQL>
```

```
SQL> ORACLE instance started.
```

```
Total System Global Area 134217728 bytes
```

```
Fixed Size 2150000 bytes
```

```
Variable Size 113127824 bytes
```

```
Database Buffers 16777216 bytes
```

```
Redo Buffers 2162688 bytes
```

Continued on next page

Database mounted.

ORA-01157: cannot identify/lock data file 4 - see DBWR trace file
ORA-01110: data file 4: 'G:\ORACLE\T99\SAPDATA3\T99_1\T99.DATA1'

SQL> Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.2.0 - 64bit Production

With the Partitioning, OLAP and Data Mining options

BR0280I BRSPACE time stamp: 2007-11-26 15.52.24

BR0279E Return code from 'g:\oracle\DEV\102\BIN\sqlplus': 0

BR0302E SQLPLUS call for database instance T99 failed

BR0306E Start and open of database instance T99 failed

BR0280I BRSPACE time stamp: 2007-11-26 15.52.24

BR0669E Cannot continue due to previous warnings or errors

BR0280I BRSPACE time stamp: 2007-11-26 15.52.24

BR0700E Fatal errors occurred - terminating processing...

BR1018I Number of instances processed: 0

BR1004E BRSPACE function 'dbstart' failed

BR1008I End of BRSPACE processing: sdwrkkax.dbr 2007-11-26 15.52.24

BR0280I BRSPACE time stamp: 2007-11-26 15.52.25

BR1007I BRSPACE terminated with errors

- c) The correct scenario to recover the database is Complete database recovery. To recover the database, start BRGUI or BRTOOLS and choose *Restore and recovery* → *Complete database recovery*. Watch the actions performed by BRRECOVER. The recovery should complete successfully without further input by pressing *Continue* at any prompt.
2. To simulate a user error, start the script usererror.bat, located in G:\oracle\<DBSID>\scripts. This script will display the current system time, which you should note. After waiting 60 seconds, the script will first display the number of rows of table *DBCHECKORA*, then drop *DBCHECKORA*. To show, that *DBCHECKORA* was dropped, the command to show the number of rows in *DBCHECKORA* is executed again, but will show an error message indicating that *DBCHECKORA* does not exist.

Which is the correct scenario to recover the database? Perform the recovery.

a)

G:\oracle\T99>cd scripts

Continued on next page

```
G:\oracle\T99\scripts>usererror.bat
This script will first display the number of table entries in
table DBCHECKORA.
Then it will drop DBCHECKORA - accessing the table after the drop
will display an error message.
For point-in-time recovery make a note of the current time which is:
04:04 PM
Script continues in 60 seconds - please wait.
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 16:05:05 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```

COUNT(*)
-----
      112

select count(*) from DBCHECKORA
                        *
ERROR at line 1:
ORA-00942: table or view does not exist
```

```
Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.2.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options
```

```
G:\oracle\T99\scripts>
```

- b) The correct recovery scenario is to perform a database point-in-time recovery. Start BRGUI or BRTOOLS and choose *Restore and recovery* → *Database point-in-time recovery*. Choose *Continue* until the BRRECOVER input menu Options for point-in-time recovery of database <DBSID> is displayed. In option End point-in-time for recovery, enter the date and time noted previously.

```
BR0657I Input menu 104 - please check/enter input values
```

```
-----
Options for point-in-time recovery of database T99
```

```
1 # Database instance of archivelog thread (instance) . []
2 ~ Last archivelog sequence to apply (last_seq) ..... []
```

Continued on next page


```

3 ~ Last system change number to apply (last_scn) ..... []
4 ~ End point-in-time for recovery (end_pit) ..... [2007-11-26
    04.04.00]

```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

- c) In the next menu, select a backup for database point-in-time recovery. Watch the actions performed by BRRECOVER. The recovery should complete successfully without further input by pressing *Continue* at any prompt.
- d) To check if table *DBCHECKORA* is back, run script *checktable.bat*.
 G:\oracle\T99\scripts>checktable.bat
 This script will display the number of table entries in
 table DBCHECKORA.

SQL*Plus: Release 10.2.0.2.0 - Production on Mon Nov 26 16:12:38 2007

Copyright (c) 1982, 2005, Oracle. All Rights Reserved.

```

COUNT(*)
-----
        112

```

Disconnected from Oracle Database 10g Enterprise Edition Release
 10.2.0.2.0 - 64bit Production
 With the Partitioning, OLAP and Data Mining options

G:\oracle\T99\scripts>

3. **(Optional)** Repeat the last exercise. Recovery will not be possible until the point-in-time of the user error. Why?
 - a) Between the latest backup and the point in time to which you want to recover the database, a database reset was performed by the previous point-in-time recovery. It is not possible to recover a database over a database reset. Therefore, you should create a complete backup immediately after any point-in-time recovery (or whole database reset).

Continued on next page

4. Because you might have messed up the database, you decide to reset the database to the point in time of the offline backup taken in the last exercise. Perform the correct scenario.
 - a) To reset the database to the point in time of a backup (offline backup or consistent online backup), use the Whole database reset scenario. Start BRGUI or BRTOOLS and choose *Restore and recovery* → *Whole database reset*.
 - b) Enter *Continue* until the list of backups suitable for whole database reset appears, and select a backup. Watch the actions performed by BRRECOVER. The recovery should complete successfully without further input by pressing *Continue* at any prompt.



Lesson Summary

You should now be able to:

- Describe a potential problem that would lead to a restore/recovery scenario
- Perform a complete recovery of the database
- Perform a point-in-time recovery of the database
- Perform a disaster restore/recovery of the database

Lesson: Advanced Backup Techniques

Lesson Overview

In this lesson advanced backup scenarios are introduced.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain the various backup strategies supported by SAP
- Decide which strategy fits your needs

Business Example

Your database contains 900 GB of data. As your users are working on the SAP system 24 hours per day from different countries, you are thinking about an alternative to normal backups that will not interrupt or slow down normal operation.

Advanced Backup Strategies

This lesson gives an overview of methods to reduce the length of the backup, restore and recovery processes, and how to ensure that backups affect live operation as little as possible. These topics are particularly relevant in systems that require high availability or have only very small windows for downtime.

Some advanced backup techniques that help to solve the following high availability problem concerning a backup strategy are discussed after this general overview. You want to perform an offline backup for a large system, but your Oracle data and SAP system always (or almost always) need to be online.

These techniques also add additional security in the case of a hardware failure.

Obviously, you have to pay for these improvements with higher costs for:

- Hardware
- Training of the administrator(s)
- Additional administration work required for the implementation and production operation

Methods for accelerating the backup and restore process

Various possibilities for optimizing backup time are illustrated below.

- Hardware

Hardware used during the backup (tape drives, disks, system and I/O buses) play a key role in the data throughput of a backup.

- Parallel backup

Using several tape drives in parallel greatly reduces the backup time.

- Using DD

If you save using BRBACKUP, do not use CPIO to copy data files. Instead use DD, which offers better performance. If you use DD to copy data, use the BRBACKUP parameter DD_FLAGS to configure the largest possible block size (for example 64K). The larger the block size, the better the general performance.

- Using the BACKINT interface

The BACKINT interface allows you to connect external backup tools to BR*TOOLS.

- Optimizing BEGIN BACKUP runtimes

Setting tablespaces in the backup mode using BEGIN BACKUP can take a long time. To minimize this time, see SAP Note 875477.

- Incremental Backup

Incremental backups are possible if you use RMAN. These backup only blocks that have been changed since the last full backup, which greatly reduces the data volume. It can also reduce runtime. However, since all blocks have to be scanned, runtime is not reduced to the same extent as the data volume. As of Oracle 10g, however, you can activate block change tracking as described in SAP Note 964619, which can save considerable runtime.

- Partial Backup

If a backup of the complete database takes too long, you can carry out several partial backups. However, you must ensure that each file of the database is included in at least one partial backup.

- Two Phase Backup

Instead of saving the database directly to tape, a quick backup to disk is performed as the first step of the two phase backup. It is then possible to save to tape in the second step.

- Split Mirror Backup

During the split mirror backup, the tablespaces are set to backup mode (online backup) or the database is stopped (offline backup). Then the mirrored disks are separated. Finally, after a very brief period, the backup mode is ended, or the database is restarted. The separated disks can now be saved to tape. To ensure a

fast restore, one or more mirrors from the last few hours or days can be stored on disk. When a restore is required, the earlier state can be restored by mounting the mirror disks without the time consuming tape method.

- Snapshots

In the snapshot method, all data from a time point is frozen at the I/O system level. When data is changed, the new version is stored as well. Creating snapshots (with the database offline or in backup mode) allows you to create backups on disk very quickly. For details on using snapshots, ask your hardware partner.

- Standby database

A standby database is created in addition to the primary database, and is recovered in real time with the data from the primary database or with a defined time offset. If required, a standby database can be started as the primary database without having to carry out a restore.

To rule out the need for a time-consuming restore from tape, several backup methods can be combined. Note, however, that you can NEVER completely rule out the need for a restore from tape, and you must ensure that a sufficient number of backup tapes are available.

Split-Mirror Disk Backup

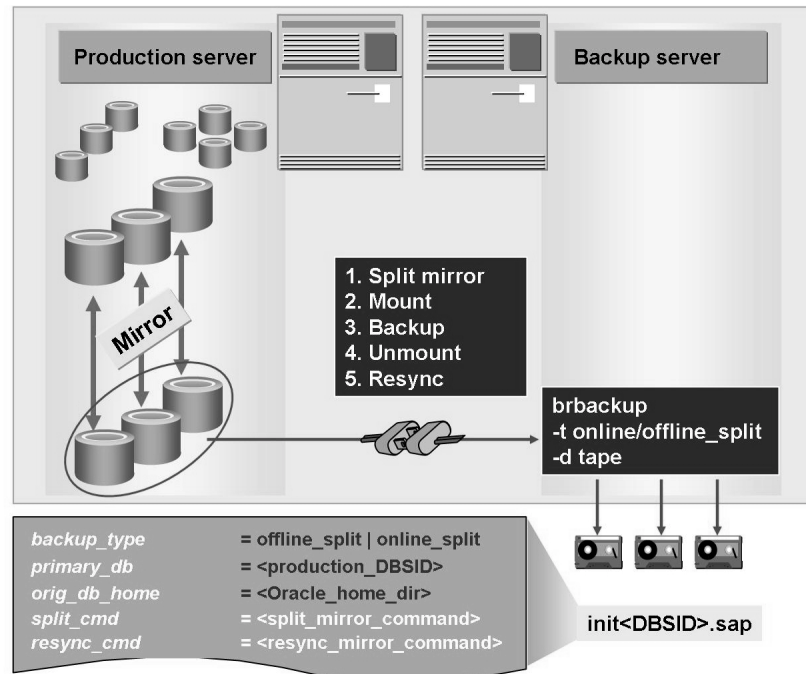


Figure 91: Split-Mirror Disk Backup

Split-mirror disk backups can significantly reduce backup time. At the start of a backup, the disk mirror where the data files are located is broken up by a predefined command. The “mirror half” is backed up from a separate server, while the “production half” is still running, without impairing performance. After the backup is finished, the disk mirror can be resynchronized immediately or with delay.

A backup is performed as follows:

Online

1. Bring the tablespaces into backup mode
2. Break up the disk mirror
3. End the backup mode in the production half
4. Perform an online backup from the mirror
5. Resynchronize the mirror

Offline

1. Stop the database
2. Break up the disk mirror
3. Start the database in the production half
4. Perform an offline backup of the mirror
5. Resynchronize the mirror

The configuration is performed by maintaining additional parameters in `init<DBSID>.sap` and must enable `BRBACKUP`, running on the backup server, to connect to the database on the production server.

During normal operation, disk mirroring protects against database failure. If such protection is also required during the backup procedure, an additional mirror is required for the production half.

With BR*Tools 7.00, you can use the Oracle Recovery Manager with split mirror backups (see SAP Note 968507). Complete backups (level 0) and incremental backups are possible in this configuration. To do this however, the directory `sapbackup` must be used by both the database and the backup server.



Caution: To perform an RMAN backup on the backup server, the database is set to the mount state by `BRBACKUP`. The Oracle software must be completely installed on the backup server.

The new parameters `pre_split_cmd` and `post_split_cmd` in BR*Tools 7.00 allow you to run external commands before and after a disk split by `BRBACKUP` (see SAP Note 968507). These commands can be executable programs or scripts.

SAP Tools and the Oracle Standby Database

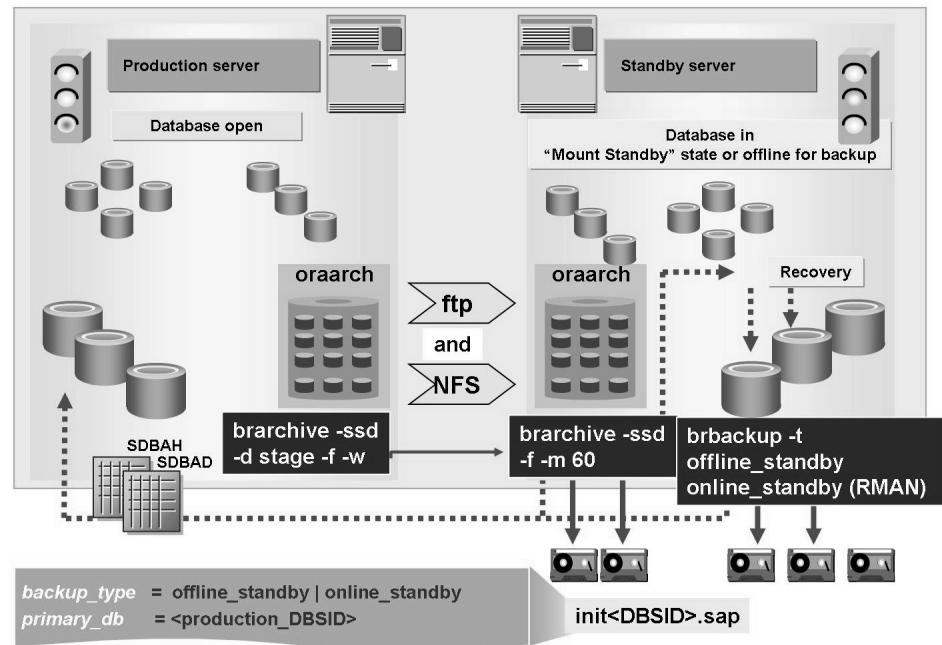


Figure 92: SAP Tools and the Oracle Standby Database

An Oracle standby database consists of two database servers. The production database has the status OPEN. During normal operation, the standby database has the status MOUNT, and is continually applying the offline redo log files from the production server. In case of a production server failure, the standby database can be opened, and can take on the role of the production database.

Data files are saved to tape on the standby server, using BRBACKUP with `offline_standby` as backup type. These actions are logged on the production server, in tables `SDBAH` and `SDBAD`, as well as in log files in directory `sapbackup` (both must be accessible from the standby server).

BRARCHIVE runs on both servers. From the production server, a continuous backup to a local, NFS mounted, or remote disk is performed (using a verification, with BRARCHIVE option `-w` | `-verify`). On the standby server, backup to tape is performed from the `oraarch` directory, which can be mounted on the standby server using Windows shares or NFS on UNIX.

The offline redo log files are applied on the standby database if you use the option `-m|-modify <delay>`. The optional entry `<delay>` determines whether the connection is “hot” (that is, replicated with no delay) or “warm” (that is, replicated with a delay). The latter makes it possible to stop applying offline redo log files before a user error is replicated on the standby server.

With BR*Tools 7.00, you can use the Oracle Recovery Manager with split mirror backups (see SAP Note 968507). This means you can perform "online" backups of standby databases without importing offline redo log files. Choose the backup type `backup_type=online_standby`. Of course, offline backups of a standby database are also possible with RMAN. Offline redo log files, however, must be imported. You can perform either complete backups (level 0) or incremental backups of the standby database.

Structure-Retaining Database Copy

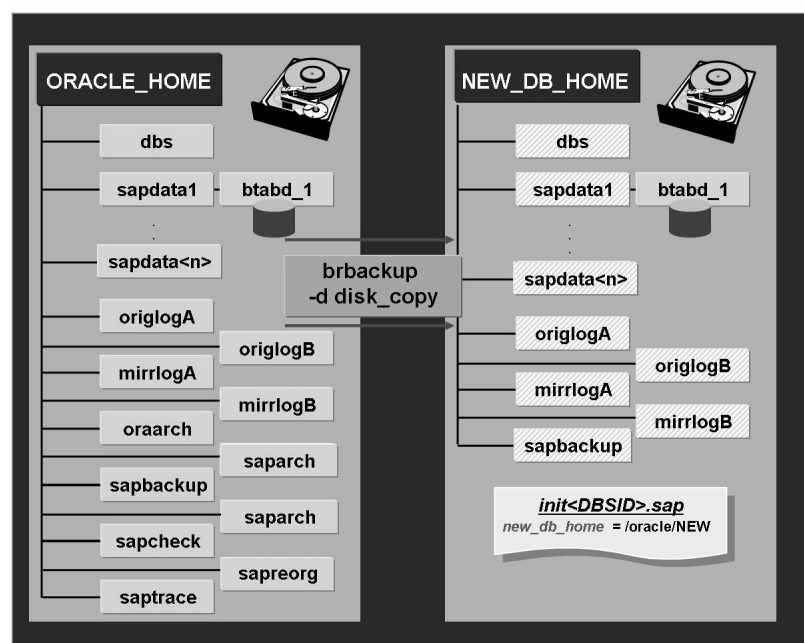


Figure 93: Structure-Retaining Database Copy

A structure-retaining database copy can be used to create a copy of the database on the same server with another `<DBSID>`, or to create a database copy on a remote server with the same or another `<DBSID>`.

A typical scenario for a structure-retaining database copy is to perform a homogenous system copy (operating system and database are equal on source and target system).

The advantage of using the structure-retaining database copy compared to a system copy performed with SAP installation tools using R3load is that this method is usually faster. On the other hand, using R3load, the database is automatically reorganized, as it first exports the data into a database and operating system-independent format and then imports the data on the new server.

System copy with BR*Tools

BR*Tools 7.00 contains important enhancements that provide additional support and automation for homogenous (within a hardware platform) and heterogeneous (between different hardware platforms) database copies. Support for homogenous database copies is based on the BRRECOVER functions "Database reset" and "Database point-in-time recovery", as opposed to the functions for heterogeneous database copies - the BRSPACE function "Recreate database" and the Oracle 10g feature "Cross-Platform Transportable Tablespaces".

Enhanced support for homogenous database copies

This new function allows a fully automated (no operator) restructuring of a database based on a complete `offline` or `online` backup, including on a computer other than the computer of the original database. You can also changed the Oracle SID (`ORACLE_SID`) and the directories (`ORACLE_HOME`) and SAPdata home (`SAPDATA_HOME`).

A database copy can be structured with BRRECOVER:

- Based on a complete/incremental offline or consistent online (`online_cons`) backup of the database (without additionally importing redo log files).
- Based on a complete/incremental online backup of the database or by additionally importing redo log files (forward recovery).

BRRECOVER and BRRESTROE automatically recognize the new environment (`ORACLE_SID`, `ORACLE_HOME`, `SAPDATA_HOME`). The database files are re-loaded to the new directories. The files are automatically renamed in the control file after the restore. At the end of the process, BRRECOVER creates new control files with the new database name.

Enhanced support for heterogeneous database copies

This new function allows you to copy an SAP Oracle database between different hardware platforms. This is an alternative to procedures that are based on Oracle export/import or SAP R3 load utilities, and can save time. These time saving advantages are reduced, however, if database files must be converted because of different Endian formats. You can also changed the Oracle SID (`ORACLE_SID`)

and the directories (ORACLE_HOME) and SAPdata home (SAPDATA_HOME). This procedure is supported as of Oracle 10g. To do this, you require the Oracle 10g feature "Cross-Platform Transportable Tablespaces".

A heterogeneous database copy is performed in the following steps:

- Export user tablespaces on the source system
- Copy the files from the user tablespaces to the target system
- Copy the scripts and dump files to the target system
- Re-create the database on the target computer
- Endian format conversion of the database files (if required)
- Import user tablespaces to the new database

If you need to convert database files owing to different endian formats, this conversion is performed by Oracle Recovery Manager (RMAN) using BR*Tools.



Hint: For more information about homogenous and heterogeneous system copies using BR*Tools, see SAP Notes 1003028.



Lesson Summary

You should now be able to:

- Explain the various backup strategies supported by SAP
- Decide which strategy fits your needs



Unit Summary

You should now be able to:

- Explain the importance of backups
- List the different backup types (offline, online, partial, and incremental backup)
- Explain the special importance of backups of the archived redo log files
- Define a backup strategy depending on database size, tape capacity, and available time for restore/recovery
- Identify the different SAP tools for backup, restore, and recovery
- Explain the concept of Oracle's Recovery Manager (RMAN)
- Customize the SAP tools
- Describe tape management with BR*Tools
- Initialize and manage backup tapes with BR*Tools
- Perform online, offline, and partial backups
- Perform RMAN backups, including incremental backups
- Create backups of archived redo log files
- Describe a potential problem that would lead to a restore/recovery scenario
- Perform a complete recovery of the database
- Perform a point-in-time recovery of the database
- Perform a disaster restore/recovery of the database
- Explain the various backup strategies supported by SAP
- Decide which strategy fits your needs



Test Your Knowledge

1. You lost data files and offline redo log files on a disk crash, because they were on the same disk. Complete recovery is possible anyway, because you perform a backup of the offline redologs with BRARCHIVE twice per day.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

2. Assuming a full database backup and all offline redo log files would fit on one tape, you would need at least _____ tape to backup the database plus offline redo log files

Fill in the blanks to complete the sentence.

3. Which of the following is true?

Choose the correct answer(s).

- ☐ A Using RMAN, you can perform online and offline backups.
- ☐ B Using RMAN, you can perform complete, incremental, and partial backups.
- ☐ C Using RMAN, you can perform whole and full backups.
- ☐ D You can perform incremental backups without using RMAN.
- ☐ E You can perform partial backups without using RMAN.
- ☐ F Before performing an incremental backup, a whole backup must be performed.

4. Before performing an incremental backup, a level ____ must be performed first. A backup of this type is called a _____.

Fill in the blanks to complete the sentence.

5. For complete recovery using the latest full and incremental backup, offline redo log files are not necessary.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

6. Concerning backups, which of the following is true?

Choose the correct answer(s).

- ☐ A When a daily offline backup is performed, backup of offline redo log files is not necessary.
- ☐ B When performing an incremental backup, an extra database block verification does not need to be performed.
- ☐ C Performing a daily consistent online backup is a sufficient backup strategy; no additional backups need to be performed.
- ☐ D An online backup also backs up the offline redo log files.

7. On a full backup, the database itself, including the Oracle software directories, are backed up.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

8. When starting a backup directly from the command line using BRBACKUP, the result and logs are also shown in transaction DB14.

Determine whether this statement is true or false.

- ☐ True
- ☐ False

9. The BR*Tools profile init<DBSID>.sap is saved by:

Choose the correct answer(s).

- ☐ A BRBACKUP
- ☐ B BRARCHIVE
- ☐ C BRTOOLS
- ☐ D Not at all

10. An offline redo log file was backed up once by BRARCHIVE. The log file now has the status _____.

Fill in the blanks to complete the sentence.

11. To perform a complete database reset, you need which of the following backups?

Choose the correct answer(s).

- ☐ A Complete offline backup
- ☐ B Complete online backup
- ☐ C Consistent online backup
- ☐ D Backup of offline redo log files

12. When you loose a disk, you can recover the database using the _____ scenario.

Fill in the blanks to complete the sentence.

13. You have a serious problem during an upgrade and decide to restore and recover the database to the state before the upgrade, that is, to the state after the SAP system was last shut down, and no other live activities were being run. As your latest backup was performed the evening before, you choose the _____ scenario.

Fill in the blanks to complete the sentence.

14. An operating system administrator accidentally deleted a data file. To recover the database, you chose the _____ scenario.

Fill in the blanks to complete the sentence.



Answers

1. You lost data files and offline redo log files on a disk crash, because they were on the same disk. Complete recovery is possible anyway, because you perform a backup of the offline redologs with BRARCHIVE twice per day.

Answer: False

The chance that you lost offline redo log files not yet backed up is high. In this case, complete recovery is not possible.

2. Assuming a full database backup and all offline redo log files would fit on one tape, you would need at least one tape to backup the database plus offline redo log files

Answer: one

Using the one-run strategy, a single tape is enough.

3. Which of the following is true?

Answer: A, B, C, E

4. Before performing an incremental backup, a level 0 must be performed first. A backup of this type is called a full backup.

Answer: 0, full backup

5. For complete recovery using the latest full and incremental backup, offline redo log files are not necessary.

Answer: False

6. Concerning backups, which of the following is true?

Answer: B

7. On a full backup, the database itself, including the Oracle software directories, are backed up.

Answer: False

8. When starting a backup directly from the command line using BRBACKUP, the result and logs are also shown in transaction DB14.

Answer: True

9. The BR*Tools profile init<DBSID>.sap is saved by:

Answer: A, B

10. An offline redo log file was backed up once by BRARCHIVE. The log file now has the status SAVED.

Answer: SAVED

11. To perform a complete database reset, you need which of the following backups?

Answer: A, C

12. When you lose a disk, you can recover the database using the Complete database recovery scenario.

Answer: Complete database recovery

13. You have a serious problem during an upgrade and decide to restore and recover the database to the state before the upgrade, that is, to the state after the SAP system was last shut down, and no other live activities were being run. As your latest backup was performed the evening before, you choose the database point-in-time recovery scenario.

Answer: database point-in-time recovery

14. An operating system administrator accidentally deleted a data file. To recover the database, you chose the Complete database recovery scenario.

Answer: Complete database recovery

Unit 3

Monitors and Tools

Unit Overview

In this unit, we will introduce monitoring and administration functionality of BR*Tools.



Unit Objectives

After completing this unit, you will be able to:

- Explain how Oracle stores its data
- Explain the difference between dictionary and locally managed tablespaces
- Explain the purpose of undo- and temporary tablespaces
- State the purpose of running regular database system checks
- Run regular checks with SAP tools
- Interpret the results of database system checks
- Create a strategy to prevent possible problems and errors
- Explain the purpose of the CCMS alert monitor
- Use the CCMS alert monitor to monitor the Oracle database

Unit Contents

Lesson: Introduction to Oracle Data Management	238
Exercise 9: Oracle Data Management	257
Lesson: Database System Check	260
Exercise 10: Database System Check	281
Lesson: CCMS Alert Monitor	288

Lesson: Introduction to Oracle Data Management

Lesson Overview

In this lesson, you will learn how Oracle stores its data.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain how Oracle stores its data
- Explain the difference between dictionary and locally managed tablespaces
- Explain the purpose of undo- and temporary tablespaces

Business Example

SAP provides tools and transactions that make monitoring and administration of an Oracle database easier, because the database administrator does not need to know exact Oracle commands to get the monitoring information and administrate the Oracle database. Nevertheless, some background is needed to interpret the monitoring results and to know which actions have to be done on the database when and why. You wish to learn more about these tools and transactions.

Tablespaces, Data Files and Segments

Oracle stores its data in tablespaces. A tablespace consists of one or more files. Data files are usually stored on the local file system, but they can also be stored on raw disks.

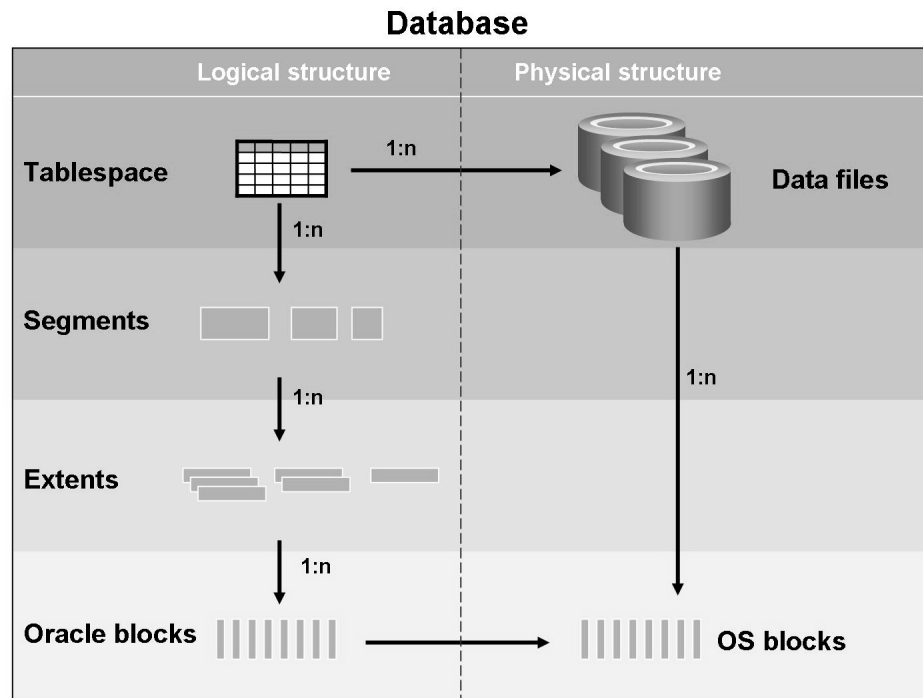


Figure 94: Tablespaces, Data Files, and Segments

Contains the actual data Oracle has four main segment types:

data

A data segment contains table data in rows.

index

Indexes are used for faster access to table data and to enforce unique constraint. Each table in SAP usually has exactly one primary index and optional secondary indexes. The index contains key fields of a table and points to the data block where the corresponding table row is stored.

temporary segment

Temporary segments are used for sorts and to create indexes.

rollback/undo segment

Rollback/undo segments are used to provide read consistency, the ability to roll back or undo changes to tables, and for recovery.

Any segment is usually stored in one single tablespace. Within the tablespace, a segment can be spread across several data files.

To meet demands of very large databases, database designers may create **partitioned tables and indexes**. Partitioned tables allow data to be split up into smaller, easily manageable units, called 'partitions'. Each partition in a table is stored in its own data segment, and can be managed individually. It can be even created in a separate tablespace (which is not recommended for SAP databases). Operations on partitioned tables and indexes can also be performed in parallel. In an SAP environment, partitioned tables are used, for example, in SAP Business Information Warehouse (SAP BW) systems (to store fact tables of InfoCubes).

This means that a single data or index segment in an Oracle database used in an SAP system holds either:

- All data for a table that is not partitioned
- All data for a partition of a partitioned table

Tablespace Naming Conventions up to SAP R/3 4.6C

Which tablespaces exist and which segments they contain depends on the SAP release on which the system was initially installed. Up to SAP R/3 4.6C, the following conventions were used:



Tablespace Naming Conventions up to SAP R/3 4.6C

Tablespace Name	Segment Type	Content
SYSTEM	tables, indexes, rollback	Oracle Data Dictionary
PSAPROLL	rollback	only rollback segments
PSAPTEMP	temp. segments	only temporary segments
PSAP<NAME>D	tables	SAP tables; tables are logically grouped into different PSAP<NAME>D tablespaces
PSAP<NAME>I	indexes	SAP indexes; all indexes to tables in a specific PSAP<NAME>D tablespace are stored in a corresponding PSAP<NAME>I tablespace

The following names are used to store SAP tables and indexes:



Tablespace names up to R/3 4.6C

Tablespace Name	Content
PSAPBTABD/I	Transaction data, frequently changed data
PSAPSTABD/I	Master data, rarely changed data
PSAPPOOLD/I	SAP pool tables
PSAPPROTD/I	Log information
PSAPLOADD/I	SAP loads (compiled ABAP programs)
PSAPSOURCED/I	SAP sources (ABAP)
PSAPDOKUD/I	Documentation tables
PSAPCLUD/I	SAP cluster data
PSAPDDICD/I	SAP data dictionary
PSAPUSER1D/I	Customer data
PSAPEL<REL>D/I	Release-dependent SAP loads
PSAPES<REL>D/I	Release-dependent SAP sources

Advantages of this tablespace layout are:

- Data files can be stored on different physical disks and controllers according to their use (PSAPBTABD, for example, has much higher I/O than PSAPES<REL>D) and segment type (tables on different disks than indexes)
- Smaller reorganizational units

Disadvantages are:

- Higher administration effort
- Less effective space management; free space in one tablespace cannot be used by any segment in another tablespace

Tablespace Naming Conventions Starting with SAP Web AS 6.10



Tablespace naming conventions starting with SAP Web AS 6.10

Tablespace Name	Segment Type	Contents
SYSTEM	tables, indexes, rollback	Oracle Data Dictionary
SYSAUX	tables, indexes	mandatory help tablespace for the SYSTEM tablespace

PSAPROLL or PSAPUNDO	rollback/undo	only rollback/undo segments
PSAPTEMP	temp. segments	only temporary segments
PSAP<SCHEMA-ID>	tables, indexes	objects of the SAP Web AS ABAP schema
PSAP<SCHEMA-ID>USR	tables, indexes	customer objects
PSAP<SCHEMA-ID><REL>	tables, indexes	release-dependent data (ABAPs, loads)
PSAP<SCHEMA-ID>DB	tables, indexes	objects of the SAP Web AS Java schema (starting with SAP Web AS 6.40)

This new tablespace layout is also called the MCODE layout (Multiple Components in One Database). Using this layout, it is now possible to store the data of several SAP systems in a single database.

To distinguish between data for SAP system C11 and system C12 stored in the same MCODE database, SAP processes no longer connect to the database with the user SAPR3, but with the user SAP<SCHEMA-ID>. Each user has its own schema (which is named like the user), and all database objects belong to the schema named for the user who created it. By default, every access to a database object accesses the object belonging to the schema of the calling user.



Note: Oracle database 10g has a new mandatory tablespace, the SYSAUX tablespace. SYSAUX is a mandatory help tablespace for the SYSTEM tablespace. It provides a central location for necessary additional meta data outside the SYSTEM tablespace. Some components and products that used the SYSTEM tablespace or their own tablespaces in earlier Oracle versions now occur in this tablespace. On the one hand, this reduces the load on the SYSTEM tablespace and on the other hand, simplifies administration. Only the actual Oracle dictionary is still located in the SYSTEM tablespace ('SYS', 'SYSTEM' schemas).

The SYSAUX tablespace is always created with new installations or database upgrades. In normal operation, the SYSAUX tablespace is neither renamed nor deleted. It can also be transported. If the SYSAUX tablespace is missing ('media failure'), certain database features will not be available or will be restricted (for example, Automatic Workload Repository (AWR), historical data, Enterprise Manager).

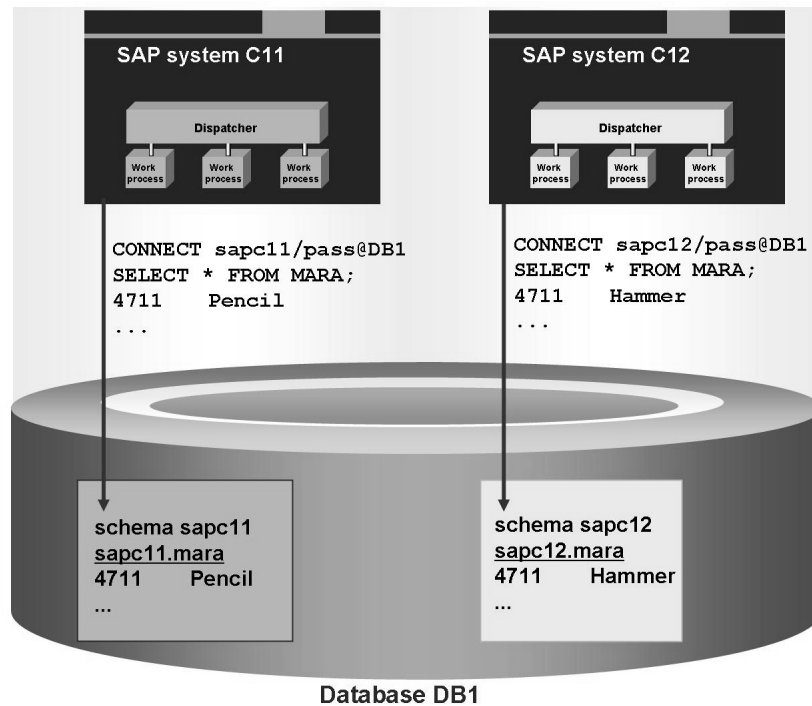


Figure 95: Database Schema

Using different schemata, SAP system C11 will, for example, see the contents of table MARA belonging to schema SAPC11 (SAPC11.MARA), because C11 connects as SAPC11. System C12 would see a completely different table (SAPC12.MARA).

While in the "old" layout, the SAP SID and the Oracle SID were always equal, in the new layout the two SIDs can be different. For the connection of the SAP instance to Oracle, a third SID-like parameter, the SCHEMA-ID, is introduced.

SAP SID (SAPSID)

System identifier for an SAP system. This is set during central instance installation and stored in the SAP profile parameter SAPSYSTEMNAME (usually in the default profile).

Database SID (DBSID)

System identifier for the database; can be different from the SAP SID. The database SID is set during database creation and can be read from the view *V\$DATABASE*, field *NAME*.

Schema ID

Identifier used as part of tablespace names (for example SAP<SCHEMA-ID>USR)) and as part of the database schema and username SAP uses to connect to the database (SAP<SCHEMA-ID>). Set by the SAP installation tool on the database installation, the schema, used by the SAP instance and various other tools like R3trans, is read from the environment variable *DBS_ORA_SCHEMA*.

All IDs **can** be different, but they **do not need** to be different:

- If MCOD is not used (every SAP system has its own database), all IDs can be identical.
- If MCOD is used, at least the second SAP system storing its data in the MCOD database needs a SCHEMA-ID different from the Database SID.

A few notes on MCOD:

- Distinguish between the MCOD tablespace layout, and have multiple components in one database. While having multiple components in one database requires the MCOD tablespace layout, having one database per SAP system does not necessarily require the MCOD tablespace layout.
- As of SAP Basis 4.6C SR2, SAP installation tools create the MCOD tablespace layout and support the load of the new data into another schema of an existing database. However, you can still load the data into a new database.
- BR*Tools still supports and will support the “old” tablespace layout in the future. Currently, there is no need to switch to the MCOD tablespace layout for existing databases (if you do not plan to use multiple components in one database).

data files

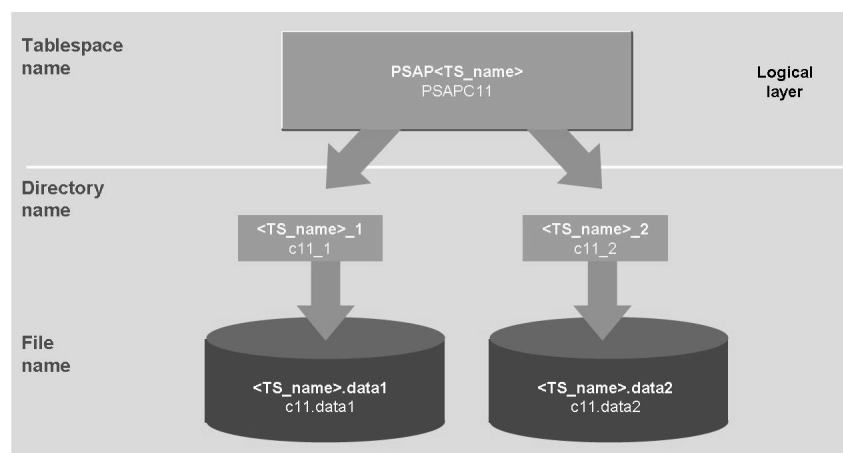


Figure 96: Naming Conventions for data files

Each tablespace consists of one or more data files that usually reside on the file system. Conventions for each data file are:

- Each data file is created in a separate subdirectory of an `sapdata<n>` directory. The directory is named like the tablespace, without the prefix *PSAP* and with a number appended that represents the `<n>`th data file of this tablespace.
- Each data file is named like the tablespace, without the prefix *PSAP* and with the extension `.data<n>`.

If a tablespace is full, there are three ways to extend the tablespace:

- Another data file can be added to the tablespace.
- An existing data file can be manually resized.
- the properties of an existing data file can be changed to be autoextensible. In this case, the file grows automatically when more space is needed in the tablespace. In this case, the maximum file size (`MAXBYTES`) and the size by which the data file is enlarged (`INCREMENT_BY`) can be specified.

There is no general recommendation as to whether the size of tablespaces should be increased manually or automatically. If tablespaces are increased manually, the free space within each tablespace must be monitored and, if there is a lack of free space, increased manually. If, on the other hand, all data files are autoextensible, the file systems have to be monitored to make sure they have enough free space for data files to grow.

While the size of a data file is less important (as there is no longer a 2 GB size limit on any supported operating system), the number of data files should be about 100 when the database has reached its expected size. So, as a rule of thumb, the data file size should be the expected database size divided by 100, but not smaller than 2 GB.

Expected database size	File size.
up to 200 GB	2 GB
200 to 400 GB	4 GB
400 to 800 GB	8 GB
larger than 800 GB	16 GB

Raw Devices

The SAP R/3 installation tools will store the Oracle data files per default on a file system. It is possible to store data files on raw devices. In this case, no file systems are created on the disks storing the data files and Oracle itself is responsible for storing its data on the raw disks. Raw devices are only supported on UNIX operating systems.

If the Oracle real application cluster (RAC) is used, the data files must either reside on a cluster file system or on raw devices. Data files on standard file systems are not supported, because several Oracle instances need concurrent write access to the data files.

Advantages of using raw devices:

- Faster I/O, because I/O is not buffered in the operating system buffer cache
- A little bit less disk space is required (missing file system overhead)
- Faster crash recovery because no file system checks have to be performed

Disadvantages of using raw devices:

- Raw devices must be well documented, as standard UNIX tools like mount or df only show filesystems, not raw devices.
- Storage configuration is inflexible because only one data file is permitted for each raw device.
- Backup is only possible using RMAN or dd (both supported by BR*Tools).

Dictionary Managed Tablespaces

All tablespaces created by SAP installation tools in the "old" tablespace layout were created as dictionary-managed tablespaces. SAP recommends using Locally Managed Tablespaces. These are the default tablespaces if the new MCOD tablespace layout is created by the installation tool.

To switch from dictionary-managed to locally managed tablespaces on existing databases, a reorganization is necessary. Whether this reorganization can be done online or must be done with export/import depends on the database version.

Dictionary-managed tablespaces and their administration is introduced to support administrators who still have to manage these tablespaces, and as a background to understand the advantages of locally managed tablespaces.

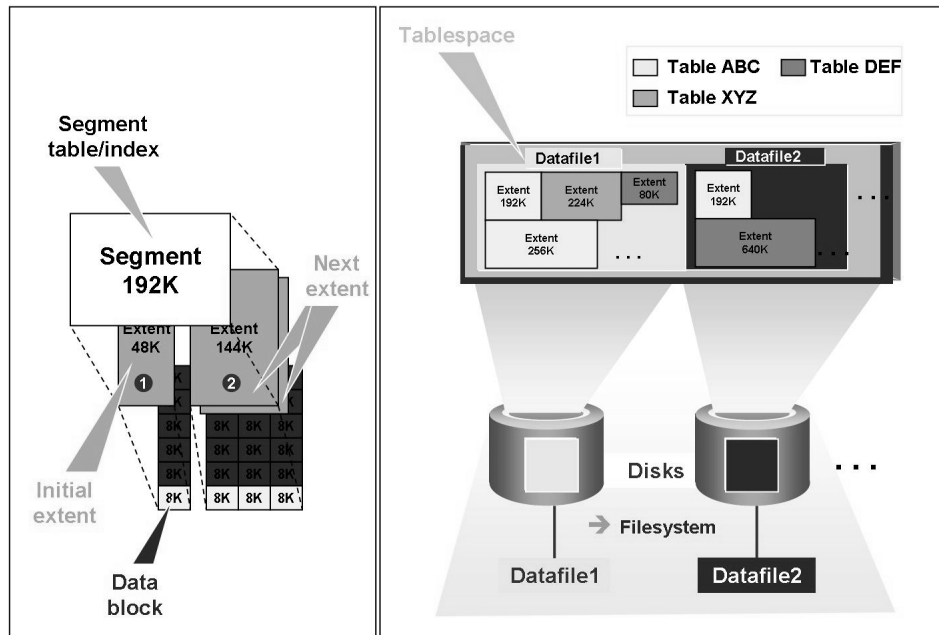


Figure 97: Dictionary Managed Tablespaces Storage Parameters

Each segment consists of one or more extents. An extent is a collection of Oracle blocks and must be allocated contiguously within a single data file.

In dictionary managed tablespaces, Oracle checks and updates data dictionary tables whenever a new extent must be allocated (for example, due to table growth) or deleted. The extent size and other parameters are called **storage parameters** and are stored in the Oracle data dictionary.

Storage parameters can be set on the creation of a tablespace. These parameters are then used for all tables in this tablespace as defaults. The defaults can be overridden when a table is created, or changed with BR*Tools for existing tables.

The parameters and their meaning are:

MINEXTENTS

Minimum number of extents allocated when a table is created. SAP installation tools usually create tables with a single initial extent; in special cases (rollback segments, reorganization, and so on) other values can be chosen.

INITIAL

Size of the initial extent which is allocated when a table is created. The SAP installation tools will create the initial extent large enough to import the table into a single initial extent if the table size is known. If the table is initially empty, the **INITIAL** parameter is calculated from information in the SAP data dictionary (tables *LAORA* and *IGORA*, maintained in transaction SE11, *technical settings* → *size category*).

NEXT

When a table grows and the existing extents are full, Oracle automatically creates a next extent with the size specified in the parameter **NEXT**.

PCTINCREASE

Not used in SAP systems (set to 0). Further extents will be created with **PCTINCREASE** with a percent larger than the previous next extent.

MAXEXTENTS

If the overall number of extents in a segment corresponds to **MAXEXTENTS**, Oracle does not create any further extents, and issues the error message ORA-01553. In SAP installations, **MAXEXTENTS** is normally set to values between 100 and 300 (see the table *IGORA*).

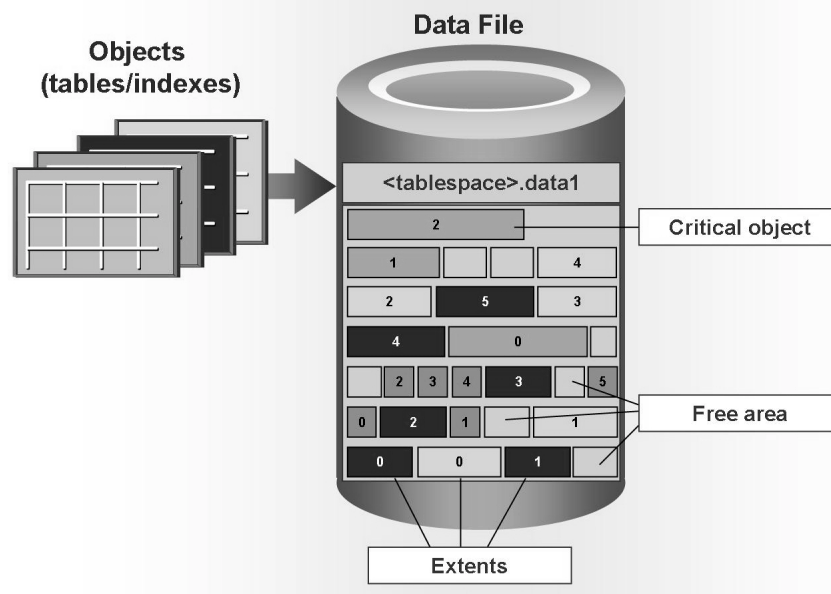


Figure 98: Dictionary Managed Tablespaces Extent Allocation

This way of allocating space for segments within tablespaces has several disadvantages and needs additional administration effort:

- Extents must be allocated contiguously within one data file. To check if a tablespace has enough free space, it is not enough to sum up all free areas. If the check then finds any segment having a NEXT size larger than the largest free area, a tablespace overflow (ORA-01653 for tables and ORA-01654 for indexes) could occur if Oracle would have to create a next extent for this segment.
- When extents are dropped, they are marked as free and their blocks can be used by new extents. But adjacent free extents are not combined into one large free extent immediately. Therefore, large free areas are not detected and will not be used for large next extents. The database administrator must coalesce free extents in this case, which means that several adjacent free extents are combined into one large free extent.
- To avoid Oracle error ORA-01553 (maxextents reached), the administrator must regularly check if any segment has a number of extents allocated that is close to MAXEXTENTS. In this case, NEXT must be increased so Oracle will create less, but larger, extents.
- Optimally, the parameter NEXT for all tables must be regularly adapted to the current table growth to make sure that Oracle does not need to create a next extent more often than, for example, one month.

The SAP tool BRCONNECT can do these checks and automatically adapt the NEXT storage parameter. The tool BRSPACE can be used to coalesce free extents manually, while `brconnect -f check` will coalesce free extents automatically. Nevertheless, using locally managed tablespaces, these problems no longer exist.

Locally Managed Tablespaces

All tablespaces created by SAP installation tools in the "new" MCODE tablespace layout are created as locally managed tablespaces.

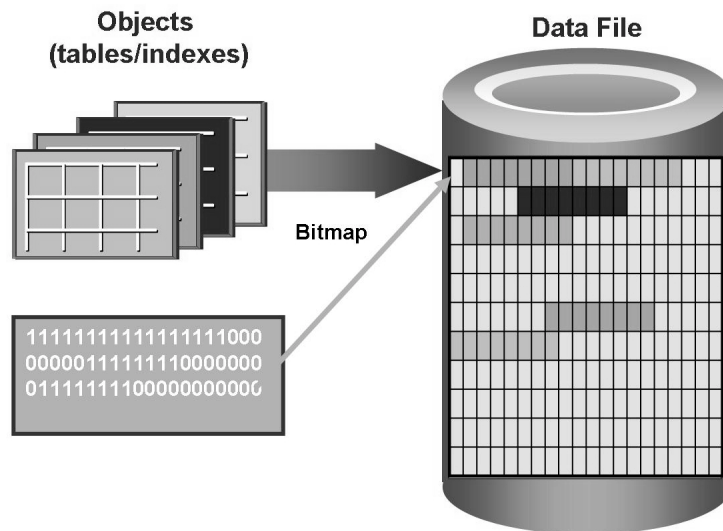


Figure 99: Locally Managed Tablespaces Extent Allocation

With locally managed tablespaces, each data file has a bitmap listing used and free blocks within the data file. When a new extent needs to be allocated, Oracle selects one data file and checks in the bitmap, whether or not enough contiguous blocks exist to create a new extent. This is done for each data file of the tablespace, for as long as there is enough free space for the allocation of the extent.

The extent size will no longer be determined by storage parameters. The extent size is either identical for all extents (UNIFORM) or automatically chosen depending on the segment size (AUTOALLOCATE). The kind of extent allocation (UNIFORM or AUTOALLOCATE) is determined at the creation of the tablespace and cannot be changed.

SAP uses locally managed tablespace with automatic extent allocation as of Oracle 9.2.0 and recommends that you create new tablespaces of this type as of Oracle 8.1.7 and SAP kernel 4.6D.

➡ **Note:** For details, see SAP Note 214995.

Exceptions are:

- The temporary tablespace PSAPTEMP is created locally managed, but with a UNIFORM extent size.
- The rollback tablespace PSAPROLL is created as dictionary-managed tablespace. For rollback, a new PSAPUNDO tablespace should replace PSAPROLL.
- In SAP BW systems, tablespaces holding fact tables or aggregates should be created with a uniform extent size of 1 MB.

For automatic extent allocation (AUTOALLOCATE), fixed rules are used to determine the size of the next extent, depending on the total size of the segment:

Segment size	Next extent size	Max. no. of extents with this size
less than 1 MB	64 kB	16
between 1 MB and 64 MB	1 MB	63
between 64 MB and 1 GB	8 MB	126
more than 1 GB	64 MB	unlimited

This fixed rule is a compromise between having too many extents and wasting disk space for segments of different sizes within tablespaces.

- A small segment with less than 1 MB will have a maximum of 16 extents; wasted space for a new extent is less than 64 kB.
- A large table with 20 GB will have 461 extents; wasted disk space due to extent allocation is less than 64 MB, which is about 0.3 % of the total used space.

Advantages using locally managed tablespaces:

- Adaption of NEXT or MAXEXTENTS is not necessary anymore; Oracle error ORA-01553 (maxextents reached) no longer occurs.
- Because the maximum extent size is 64 MB, less fragmentation of a data file occurs and the available space within a data file is better used.
- Better performance when dropping or creating tables and on parallel loads, because no data dictionary tables have to be updated when new extents are created or deleted.

Disadvantage using locally managed tablespaces:

- Check for used and free space is more expensive

Rollback and Undo Tablespace

Besides storing data, the Oracle database must provide:

Transactional Consistency

Normally, an individual database transaction performs inserts, updates or deletes in many different tables. On SAP systems, these database transactions are typically performed by the update and background work processes. If such a database transaction terminates before all changes were performed in the database and the work process commits its changes, Oracle has to be able to rollback or undo these changes.

Read Consistency

It often happens that one transaction reads data from the database while it is actually being changed or deleted by another transaction. While the changing transaction is not yet finished, any reading process must get the information existing before the changing transaction has modified the data.

To provide transactional consistency and read consistency, Oracle uses rollback segments in a rollback tablespace or, as of Oracle 9i, undo segments in an undo tablespace.

When a database transaction starts, Oracle selects one existing rollback or undo segment for this transaction. Whenever table data is changed, the data is changed in its original block, after the old data was copied into the assigned rollback or undo segment. Whenever a rollback is necessary (for example, a transaction was canceled or an instance recovery is performed after an instance crash), the original data will be recovered from the rollback or undo segments (transactional consistency). Whenever a process reads data that is actually changed by another process, the reading process reads from the rollback or undo segments (read consistency).



Hint: The SAP installation tools up to Web AS 6.20 create a rollback tablespace. SAP recommends to switch to the undo tablespace when using Oracle 9i, because the undo tablespace is easier to handle and provides some new functionality.

Rollback Tablespace

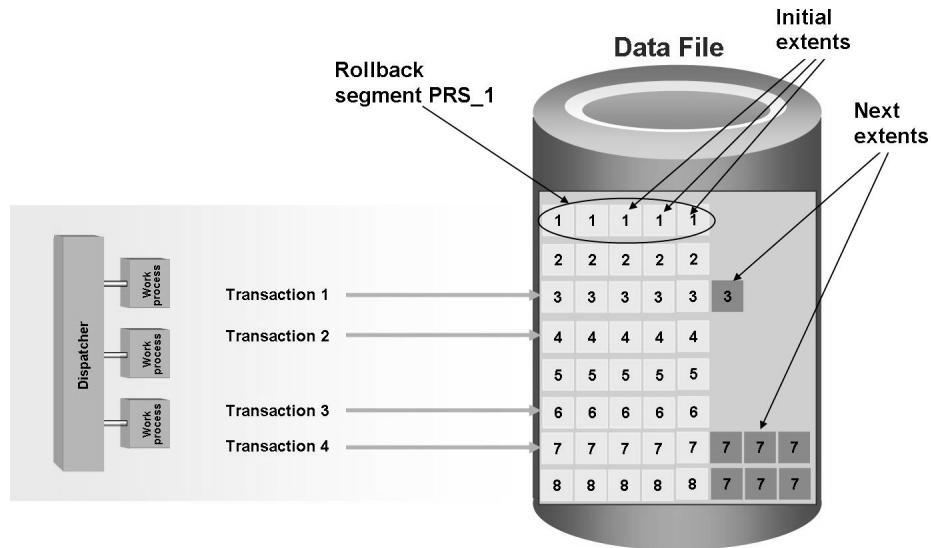


Figure 100: Rollback Segments and Extents

The internal extent management of the rollback tablespace is very similar to the internal extent management of dictionary-managed tablespaces. Rollback segments are created by the SAP installation tool using the storage parameters `INITIAL`, `MINEXTENT`, `MAXEXTENT`, and `NEXT`.

For rollback segments, the special storage parameter `OPTIMAL` is set. Whenever a rollback segment is fully occupied, Oracle will automatically allocate a `NEXT` extent. When any transaction using this rollback segment commits, the freed extents are marked to be overwritable by other transactions. In certain cases, Oracle will also try to regain free space within the rollback tablespace by shrinking the rollback segments to their `OPTIMAL` size.

A common error caused by this behavior is ORA-01555 (snapshot too old):

1. One process (for example a background job) starts changing or deleting data.
2. Another process reads the data being changed by the other process.
3. The changing process commits because the commit within the rollback segment extents are overwritten by a third process, or the rollback segments shrinks to its optimal size.
4. The reading process cannot read anymore from the rollback segments: ORA-01555.

As with data dictionary tablespaces, this way of managing extents has several disadvantages and needs additional administrative effort:

- Setting good storage parameters for the rollback segments depends on database size and on which actions are performed in the SAP system. Often, experimenting with the parameters is necessary.
- Because Oracle can select any rollback segment for a transaction, storage parameters must always be changed for all rollback segments.
- For special actions like large client copies, the storage parameters used for normal operation are not good. In this case, SAP recommends to switch to a special, larger rollback tablespace (usually called PSAPROLLBIG) with special storage parameter settings. After this special action, a switch back to the standard rollback segments must be performed.

Undo Tablespace

As of Oracle 9i, Oracle introduced the new feature called automatic undo management (AUM). This feature uses a new tablespace type called undo tablespace (SAP convention: PSAPUNDO) with undo segments instead of rollback segments.

There are currently no disadvantages known when using automatic undo management. The advantages are:

- Undo segments are managed automatically. There are no storage parameters to maintain; even the number of undo segments is configured automatically and can change dynamically.
- One transaction can use more than one undo segment.
- The criteria for undo segments being overwritten is no longer the commit of the transaction, but a time period defined by the profile parameter `undo_retention`. Setting the undo retention time equal to the runtime of the longest running background job would avoid ORA-01555 completely (as long as the undo tablespace is large enough).
- The conversion to larger undo tablespaces for special actions such as large client copies is more simple.

For automatic undo management, four new profile parameters are available:

undo_management

This parameter can be set to `MANUAL` or `AUTO`. To turn automatic undo management on, the parameter must be set to `AUTO`.

undo_tablespace

The undo tablespace is a special tablespace created with the Oracle command `CREATE UNDO TABLESPACE 'PSAPUNDO'`. The name of the undo tablespace (usually PSAPUNDO) must be specified in this parameter.

undo_retention

This is the time, in seconds, that Oracle keeps undo data (called undo retention time). After this time period, undo data can be overwritten.

undo_suppress_errors

When activating automatic undo management, manual management of rollback segments is no longer possible and would cause errors. To suppress errors concerning manual management of rollback segments (for example in scripts), this parameter can be set to *true*.



Caution: This procedure is no longer supported as of Oracle 10g.



Hint: SAP generally recommends that you switch from rollback tablespace to undo tablespace as of Oracle 9.2.0. The procedure to switch is described in detail in SAP Note 600141; here, only a general description is given.

To switch from manual to automatic undo management, the following steps generally have to be performed:

1. Create the new undo tablespace PSAPUNDO.
2. Change or insert the four new Oracle parameters with appropriate values.
3. Drop all rollback segments (except the SYSTEM rollback segment).
4. Delete (or comment out) the Oracle parameter `rollback_segments`.
5. Drop the rollback tablespace.

Exercise 9: Oracle Data Management

Exercise Objectives

After completing this exercise, you will be able to:

- Identify tablespace types

Business Example

You want to check whether your database still uses dictionary-managed tablespaces and a rollback tablespace.

Task:

Check the tablespace types of your database.

1. Check which tablespace are dictionary-managed and which are locally managed.
Check if your database uses a rollback or undo tablespace.

Solution 9: Oracle Data Management

Task:

Check the tablespace types of your database.

1. Check which tablespace are dictionary-managed and which are locally managed.
Check if your database uses a rollback or undo tablespace.

- a) To check the tablespace types, start BRGUI or BRTOOLS and choose *Space management* → *Additional space functions* → *Show tablespaces*.
The list shows:

List of database tablespaces

Pos.	Tablespace	Type	Status	ExtMan.	SegMan.	Backup	Files/AuExt.
	Total [KB]	Used[%]	Free [KB]	MaxSize [KB]	ExtSize [KB]		FreeExt.
	Largest [KB]						
1 -	PSAPT99	DATA	ONLINE	LOCAL	AUTO	NO	1/1
	20480	16.88	17024	51200	30720		1
	30720+:17024:0:0:0						
2 -	PSAPT99USR	DATA	ONLINE	LOCAL	AUTO	NO	1/0
	10240	0.63	10176	10240	0		1
	10176:0:0:0:0						
3 -	PSAPTEMP	TEMP	ONLINE	LOCAL	MANUAL	NO	1/0
	20480	0.00	20480	20480	0		0
	0:0:0:0:0						
4 -	PSAPUNDO	UNDO	ONLINE	LOCAL	MANUAL	NO	1/0
	20480	0.31	20416	20480	0		229
	1024:1024:1024:1024:1024						
5 -	SYSAUX	DATA	ONLINE	LOCAL	AUTO	NO	1/0
	204800	29.72	143936	204800	0		1
	143936:0:0:0:0						
6 -	SYSTEM	DATA	ONLINE	DICT	MANUAL	NO	1/0
	409600	44.51	227304	409600	0		1
	227304:0:0:0:0						

- b) You have a tablespace called PSAPROLL, so a rollback tablespace instead of an undo tablespace is used.
- c) PSAPT00USR and SYSTEM are dictionary-managed tablespaces.



Lesson Summary

You should now be able to:

- Explain how Oracle stores its data
- Explain the difference between dictionary and locally managed tablespaces
- Explain the purpose of undo- and temporary tablespaces

Lesson: Database System Check

Lesson Overview

In this lesson, you will learn how to use and customize the database system checks.



Lesson Objectives

After completing this lesson, you will be able to:

- State the purpose of running regular database system checks
- Run regular checks with SAP tools
- Interpret the results of database system checks
- Create a strategy to prevent possible problems and errors

Business Example

Until last Thursday, your SAP system worked fine, but in the morning the phone didn't stop ringing. One user told you that she received an ABAP short dump indicating a broken update, other users were complaining that they didn't see the data they entered shortly before, and others report broken background jobs.

You checked the ABAP short dump, which indicated Oracle error ORA-01653. After checking the Oracle alert log file, you found the same error, saying that a tablespace is full and needs to be enlarged.

If you had run the regular checks introduced in this lesson, you would have been warned two weeks ago that the tablespace was using more than 95 %. You had the chance to enlarge the tablespace before the tablespace overflow occurred.

Checking the Database



DBA Cockpit - DB13

- ☒ **Check the database**
- ☐ **Adapt next extents**
- ☐ **Check and update statistics**
- ☐ **Cleanup logs**

Figure 101: BRCONNECT Activities: Check Database

BRCONNECT is the main tool used to check the database. Besides checking the database, BRCONNECT can also adapt the NEXT extent size, create the optimizer statistics, and clean up DBA logs.

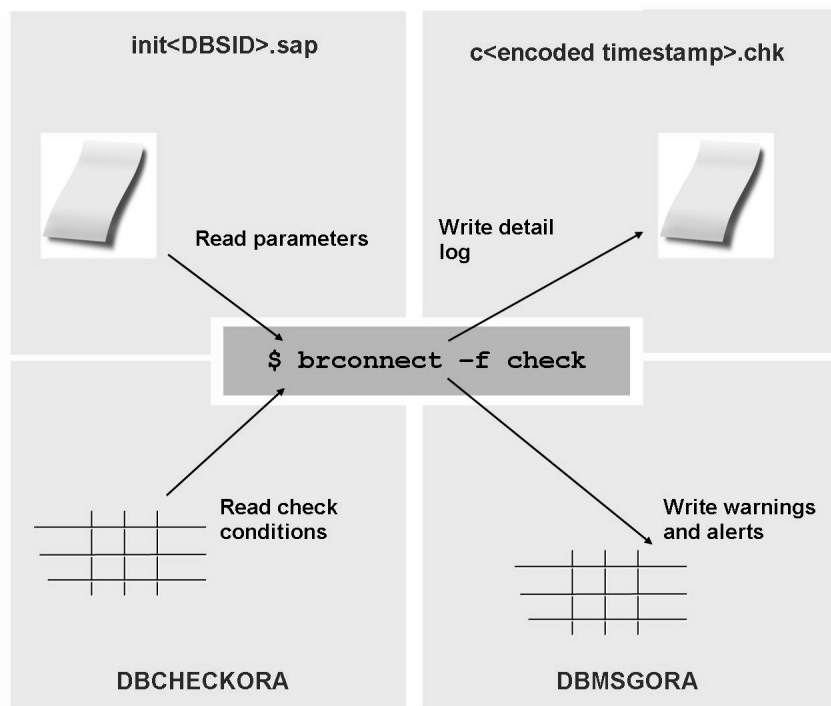


Figure 102: brconnect -f check

The tool `brconnect -f check` should be scheduled daily.

The default behavior of BRCONNECT can be adapted with parameters in the file `$ORACLE_HOME/dbs/init<DBSID>.sap` in UNIX or `%ORACLE_HOME%\database\init<DBSID>.sap` in Windows. To override these defaults, command line options can be used.

To check the database, BRCONNECT reads which checks it should perform, as well as conditions for the checks, from the table `DBCHECKORA`.

If any warnings or errors are detected, they are written into the table `DBMSGORA` and a detailed log file, `$SAPDATA_HOME/sapcheck/c<encoded timestamp>.chk`, is created.

➡ **Note:** Normally, BRCONNECT is started as follows: `brconnect -u / -c -f check`.

Parameters for the Database System Check

To perform the database system check, start BRGUI or BRTOOLS and choose *Check and verification* → *Database system check*. Alternatively, run the command `brconnect -f check` in the command line. The following table lists command line options for `brconnect -f check` (whose long names are identical to the input menus of BRTOOLS) and the corresponding parameters in `init<DBSID>.sap`.



Parameters for the Database System Check

Option	Parameter	Meaning
<code>-o -owner</code>	<code>check_owner</code>	By default, BRCONNECT checks objects of all SAP owners. Use this to restrict the checks to objects of certain owners, for example in MCOB databases.
<code>-e -exclude</code>	<code>check_exclude</code>	Per default, BRCONNECT checks all objects. Use this to exclude checks on certain tables. Possible arguments are: <ul style="list-style-type: none"> • <code><tables></code> • <code><indexes></code> • <code><tablespaces></code> • <code>all_part</code> to exclude non SAP partitions (such as in SAP BW and SAP APO) • <code>non_sap</code> to exclude non SAP objects (such as Oracle data dictionary tables)
<code>-d -default</code>		Normally, check conditions are read from the table <code>DBCHECKORA</code> . Use this option so that BRCONNECT performs the check with the default conditions in the source code.

Check Conditions for BRCONNECT

BRCONNECT reads the checks to be performed from the table `DBCHECKORA`. To adapt the checks and their conditions, table `DBCHECKORA` can be customized using transaction DB17.



Typ	Condition	Object	Activ	Level	Oper	Val	Unit	Period	Unit	Date	User	Corr	Corr. Measure
DBA	ARCHIVER_STUCK		<input checked="" type="checkbox"/>	W	>	90	P					D	Save and delete archive log files
DBA	CONTROL_FILE_MIRROR		<input checked="" type="checkbox"/>	E								D	Extend CONTROL_FILES parameter
DBA	CONTROL_FILE_MISSING		<input checked="" type="checkbox"/>	E								D	Restore the file
DBA	CRITICAL_FILE		<input checked="" type="checkbox"/>	W								D	Extend the file system or change autoextend parameters
DBA	CRITICAL_SEGMENT		<input checked="" type="checkbox"/>	W	<=	2						D	Extend the tablespace
DBA	CRITICAL_TABLESPACE		<input checked="" type="checkbox"/>	W								D	Extend the file system or change autoextend parameters
DBA	DATA_FILE_MISMATCH		<input checked="" type="checkbox"/>	E								D	Restore the file
DBA	DATA_FILE_MISSING		<input checked="" type="checkbox"/>	E								D	Restore the file
DBA	FILE_OFFLINE		<input checked="" type="checkbox"/>	E								D	Set the database file online
DBA	FILE_SYSTEM_FULL		<input checked="" type="checkbox"/>	W	>	99	P					D	Extend the file system
DBA	HARMFUL_STATISTICS		<input checked="" type="checkbox"/>	E								D	Delete optimizer statistics
DBA	INVALID_FILE_TYPE		<input checked="" type="checkbox"/>	E								D	Check the file
DBA	IN_WRONG_TABLESPACE		<input checked="" type="checkbox"/>	W								D	Move table or index to the right tablespace
DBA	MISSING_INDEX		<input checked="" type="checkbox"/>	E								D	Create an index
DBA	MISSING_STATISTICS		<input checked="" type="checkbox"/>	E								D	Collect optimizer statistics
DBA	NOARCHIVELOG_MODE		<input checked="" type="checkbox"/>	E								D	Set the database in ARCHIVELOG mode
DBA	PCTINCREASE_NOT_ZERO		<input checked="" type="checkbox"/>	W								D	Set PCTINCREASE value to zero
DBA	REDOLOG_FILE_MIRROR		<input checked="" type="checkbox"/>	E								D	Add a redo log group member
DBA	REDOLOG_FILE_MISSING		<input checked="" type="checkbox"/>	E								D	Restore the file
DBA	TABLESPACE_FULL		<input checked="" type="checkbox"/>	W	>	95	P					D	Extend the tablespace
DBA	TABLESPACE_IN_BACKUP		<input checked="" type="checkbox"/>	W								D	Set the tablespace out of backup mode
DBA	TABLESPACE_OFFLINE		<input checked="" type="checkbox"/>	E								D	Set the tablespace online
DBA	TOO_MANY_EXTENTS		<input checked="" type="checkbox"/>	W	>	90	P					D	Increase MAXEXTENTS of the segment
DBO	ARCHIVE_TOO_OLD		<input checked="" type="checkbox"/>	W	>	10	D					D	Start an archive log backup
DBO	BACKUP_TOO_OLD		<input checked="" type="checkbox"/>	W	>	10	D					D	Start a complete database backup
DBO	LAST_ARCHIVE_FAILED		<input checked="" type="checkbox"/>	W								D	Check the archive log backup log
DBO	LAST_BACKUP_FAILED		<input checked="" type="checkbox"/>	W								D	Check the database backup log
DBO	LAST_OPERATION_FAILED		<input checked="" type="checkbox"/>	W								D	Check the operation log
DBO	LAST_OPERATION_FAILED	chk	<input checked="" type="checkbox"/>	W								D	Check the operation log
DBO	LAST_STATS_FAILED		<input checked="" type="checkbox"/>	W								D	Check the update optimizer statistics log

Figure 103: Customizing *DBCHECKORA* with DB17

Using transaction DB17, you can:

- Add new conditions of type DBO, ORA, or PROF
- Exclude individual conditions from the check
- Specify threshold values for the conditions
- Create object-specific conditions to exclude them from the check
- Create object-specific conditions to set individual threshold values
- Specify corresponding corrective actions (this field is only for information and is displayed when the error occurred)
- Maintain the description of the conditions

In *DBCHECKORA*, there are four different type of checks defined:

Database administration (DBA)

Database administration checks, like database configuration, space management, and state of the database. Checks of this type cannot be added, as the actions these checks perform are hardcoded into BRCONNECT. Nevertheless, you can copy existing check conditions and so specify special thresholds for specific tablespaces specified in the *OBJECT* field. To exclude objects from the check, use the *check_exclude* parameter in *init<DBSID>.sap*. To exclude whole checks, deactivate them using transaction DB17.



Note: For more information about possible test operands, threshold values, and value units, see SAP note 435290.

Database operation (DBO)

Database operation checks, like backup results and failed operations. Here you can create new *LAST_OPERATION_FAILED* and *LAST_OPERATION_TOO_OLD* check conditions for any BR*Tools function ID. This allows you to monitor other BR*Tools operations.

Oracle messages (ORA)

In this part of the check, the alert log file is searched for specific Oracle error messages. Additional checks for other Oracle messages in the alert log file can be added.

When using transaction DB17 to define additional checks in the alert log file, all entries are automatically converted into uppercase letters. To include checks for strings that contain lowercase letters, this additional entry must be inserted with SQL*Plus.



Hint: See the online help for a detailed description of the SQL command.

Oracle profile parameters (PROF)

Here, Oracle profile parameters are checked. These checks can be adapted to the current recommendations from SAP, and new checks can be included.



Hint: Current recommendations for Oracle parameters can be found in SAP Note 124361 for SAP R/3 systems, and SAP Note 180605 for SAP BW systems.

Viewing the Output of the Database System Check

The detailed log of the database system check is written into the directory \$SAPDATA_HOME/sapcheck (UNIX) or %SAPDATA_HOME%\sapcheck (Windows). The filename is an encoded timestamp and has the extension .chk.

The detailed log can be viewed with any text viewer or editor. In addition, BRTOOLS can be used to display the logfiles. Call BRTOOLS or BRGUI and choose *Additional functions* → *Show profiles and logs* → *BRCONNECT logs*.



Display of BRCONNECT Logs

BR0658I List menu 52 - please select one entry

Display of BRCONNECT logs

Pos.	Log	Start	Function	Rc	Object
1	= cdwroebt.chk	2007-11-27 10.16.45	check	1	
2	- cdwroebh.cln	2007-11-27 10.16.33	cleanup	0	
3	- cdwroeao.sta	2007-11-27 10.16.14	stats	0	ALL
4	- cdwrodki.chk	2007-11-27 10.09.12	check	1	
5	- cdwroczu.chk	2007-11-27 10.04.38	check	1	
6	- cdwrocuz.next	2007-11-27 10.02.33	next	0	ALL
7	- cdwrobht.chk	2007-11-27 09.45.33	check	1	
8	- cdwrjngl.chk	2007-11-26 11.35.31	check	1	
9	- cdwrjmgk.chk	2007-11-26 11.24.14	check	3	
10	- cdwrjmco.chk	2007-11-26 11.22.34	check	3	
11	- connT99.log	<summary log>			

To view the BRCONNECT logs from within the SAP system, transaction DB14 can be used. Viewing the logs with DB14 has the advantage that warnings and errors are automatically highlighted.

In addition to the output in the log file, the database system check also writes warnings and errors into the database table *DBMSGORA*. This table can be viewed with transaction DB16 and is also read by the CCMS alert monitor (transaction RZ20) to check and report Oracle database alerts.

Interpreting the Detail Log of the Database System Check

The detail log of the database system check first contains information about the brconnect parameters, then information about the database objects:

Tablespaces and data files

A list of all tablespaces and their data files. This list shows the status and the current size of all data files.

- A plus sign (+) behind the status of a tablespace indicates that the corresponding tablespace is locally managed. A pound sign (#) behind the status of a tablespace indicates that this is a temporary tablespace. A minus sign (-) behind the status of a tablespace indicates that this is an undo tablespace.
- A plus sign (+) behind the status of a data file indicates that the corresponding data file is autoextensible.

Redo Log Files

A list of all redo log files, their status, and their size

Control files

A list of control files and their size

Database disk volumes

A list of disk volumes containing database data. For each main directory, the total and free space is shown, including the percentage of used space.

Tablespace fragmentation

First, for each tablespace, statistical information like number of data files (*Files*), tables, indexes, and extents is shown, as well as **current** total, used, and free space within the tablespace.

The next columns, *MaxSize[KB]*, *Used[%]*, and *Free[KB]*, show the information about the tablespaces, keeping the auto extensibility of data files in mind.

This information is identical to the current sizes for tablespaces not having autoextensible data files. For tablespaces having autoextensible data files (indicated by a plus sign [+] behind the numbers), the numbers show:

- Space information, assuming each data file has enough disk space to grow up to its `MAX_SIZE` value
- Maximum available disk space, if the disk space available smaller than `MAX_SIZE`. In this case, further down, a *CRITICAL_FILE* alert is raised, as a file system overflow could occur if Oracle would extend the data file.

The last column, *Largest[KB]*, lists the five largest extents. A plus sign (+) indicates again that this size would be available if a data file would be automatically extended.

Check Conditions for Database Administration

Lists all check conditions from *DBCHECKORA*, including their threshold values, descriptions, and severity. It is also indicated if the check is active or not.

The detail log ends with a list of all warnings, errors, or alerts detected and a summary information.

Adapting the NEXT Extent Size



DBA Cockpit - DB13

- ☐ Check the database
- ☒ Adapt next extents
- ☐ Check and update statistics
- ☐ Cleanup logs

Figure 104: BRCONNECT Activities: Adapt nNext Extents

In dictionary-managed tablespaces, the storage parameter NEXT determines the size of the next extent when a new extent must be allocated.

The tool `brconnect -f next` should be executed regularly (for example once per week) to adapt the NEXT storage parameter of all segments in dictionary managed tablespaces. This will prevent the following:

- Avoid tables having too many small extents, which would:
 - Negatively affect performance
 - Increase the chance that the total number of extents reaches the limit `MAXEXTENTS`
- Avoid tables having too large NEXT extent size, which would:
 - Waste diskpace
 - Cause fragmentation as smaller contiguous free areas cannot be used
 - Cause a tablespace overflow if the largest free area is smaller than the NEXT extent size of a table



Hint: Per default, `brconnect -f next` checks for all tables and indexes if the NEXT storage parameter has to be adapted. All objects in locally managed tablespaces are automatically excluded. It is not necessary to manually restrict it to objects in dictionary managed tablespaces.



Hint: Run `brconnect -f next` once to check if there are tables in data dictionary tablespaces available to adapt the NEXT extent size. If the message `BR0906I All selected tables/indexes are in locally managed tablespaces` is shown, there is no need to schedule `brconnect -f next`



Note: Normally, `BRCONNECT` is started as follows: `brconnect -u / -c -f next -t all`.

The algorithm used to determine the optimal NEXT extent size makes sure that the NEXT extent size is:

- Never changed to a value smaller than the current value and as specified in the SAP data dictionary (*TGORA* for tables and *IGORA* for indexes) for the corresponding size category of this table or index (stored in the SAP data dictionary table *DD09L*)
- About 10 % of the total size of the table or index, if the table is larger than its expected size
- Smaller than the largest contiguous free area of the tablespace, if the tablespace is not autoextensible

In most cases it is not necessary to specify additional parameters to `brconnect -f next`. On the other hand, sometimes exceptions (for example exclusion of tables) or special settings (for example, forcing a specific NEXT extent size for specific tables) are necessary. These settings can be done on the command line, or permanently by setting parameters in the profile `init<DBSID>.sap`:



Adapt Next Extents: Optional Parameters

Option	Parameters	Meaning
<code>-e -exclude</code>	<code>next_exclude</code>	List of tables or indexes for which the NEXT extent is not adapted. The keyword <code>all_part</code> excludes SAP partitions (such as in SAP BW and SAP APO).
<code>-s -special</code>	<code>next_special</code>	Defines special settings for a list of tables or indexes. Syntax: <code><object>:<size>[/<limit>]</code> , in which: <ul style="list-style-type: none"> • <code><object></code> is a table or index • <code><size></code> is NEXT extent size for the <code><object></code> • <code><limit></code> is optional and specifies the MAXEXTENTS value for the <code><object></code>
<code>-m -max</code>	<code>next_max_size</code>	Defines the maximum size for any NEXT extent. No extent will be adapted to be larger than this value
<code>-l -limit</code>	<code>next_limit_count</code>	Defines the maximum number of extents of a segment (MAXEXTENTS).

Option	Parameters	Meaning
-f -force		<p>Forces the next extent size to be reduced. The following argument defines the target NEXT size for the reduction:</p> <ul style="list-style-type: none"> • <code>free</code>: reduces the NEXT extent size to the size of largest contiguous free area in the tablespace. • <code>max</code>: reduces the NEXT extent size to the the value of <code>next_max_size</code>. • <code>both</code> (default) reduces the NEXT extent size to the smaller of the values <code>free</code> and <code>max</code>.
-t -table	<code>next_table</code>	List of tables, indexes or tablespaces to adapt the next extent; default: all tablespaces. Keyword <code>special</code> adapts only objects specified in the <code>next_special</code> parameter.
-o -owner	<code>next_owner</code>	Limits the adjustment to tables owned by certain owners; default: tables of all SAPR3 and SAP<SCHEMA-ID> users.

The detailed log of `brconnect -f next` is written into the directory `$SAPDATA_HOME/sapcheck` (UNIX) or `%SAPDATA_HOME%\sapcheck` (Windows). The file name is an encoded timestamp and has the extension `.nxt`. It can be viewed in the same way as the log of the database system check.

Checking and Updating Database Statistics



DBA Cockpit - DB13

- ☐ Check the database
- ☐ Adapt next extents
- ☒ Check and update statistics
- ☐ Cleanup logs

Figure 105: BRCONNECT Activities: Check and Update Optimizer Statistics

Oracle uses the cost-based optimizer to find the optimal access path of an SQL query. The cost-based optimizer relies on statistics about tables, which are stored in Oracle dictionary tables in the SYSTEM tablespace. If these statistics are not up to date, for example because a table grew very much, Oracle might use a more expensive access path when querying tables, which will result in poor performance.

On SAP systems, the parameter `optimizer_mode` is set to `CHOOSE`. This means that Oracle will use the cost-based optimizer when statistics are available. If no statistics are available, they are created when the table is accessed at runtime.

To update the statistics on an SAP system, use the tool `brconnect -f stats`. This tool will check all tables of the database to see if the statistics are out of date and will update the statistics if required.



Caution: Do not use other tools to update the statistics, as these tools might not be aware of the SAP tables that must not have statistics!

To make sure that statistics are up to date, SAP recommends that you run `brconnect -f stats -t all` regularly. It can be scheduled in the DBA Cockpit (transaction DB13: check and update statistics). With Oracle 10, schedule the statistics run once a week. Since table monitoring is active for all tables by default, the check phase is very quick and does not require that much time.



Hint: With older Oracle releases (8.1.7 and 9i), table monitoring is not active. During the statistics run, the check phase requires considerable time. The statistics are therefore scheduled once a week.



Hint: Up to SAP R/3 4.6C transaction DB13, by default, does not offer a single option to check and update the statistics in one run. You have to schedule two jobs: The statistics are first checked, and then updated. To change to the new one-phase strategy in SAP R/3 4.6C, apply the changes described in SAP Note 403704.

Run `brconnect -f stats -t missing` whenever new tables are created that do not have statistics. It can be scheduled with transaction DB20 (choose *Global statistics* → *Create missing*). Such tables will be reported by the database system check. Missing statistics will be also collected in the standard run `brconnect -f stats -t all`.



Note: Normally, BRCONNECT is started as follows: `brconnect -u / -c -f stats -t all`.

The detailed log of `brconnect -f stats` is written into the directory `$SAPDATA_HOME/sapcheck` (UNIX) or `%SAPDATA_HOME%\sapcheck` (Windows). The file name is an encoded timestamp and has the extension `.sta`. It can be viewed in the same way as the log of the database system check.

Clean up old logs and traces



DBA Cockpit - DB13

- ☐ Check the database
- ☐ Adapt next extents
- ☐ Check and update statistics
- ☒ Cleanup logs

Figure 106: BRCONNECT Activities: Cleanup Logs

Over time, many log files and other intermediate files, such as backups on disks, are created. To clean up these logs, use `brconnect -f cleanup`. This process cleans:

- Detailed logs of the BR*Tools
- Disk backups created by BRBACKUP and BRARCHIVE
- Export dumps and scripts created by BRSPACE
- Log records and check results in the tables *SDBAH*, *SDBAD*, *DBA**, and *DBMSGORA*
- Oracle trace and audit files

➡ **Note:** Normally, BRCONNECT is started as follows: `brconnect -u / -c -f cleanup`.

By default, all files older than 30 days are cleaned up, and database log records older than 100 days are deleted. You can define another cleanup period for each object separately in the BR*Tools profile `init<DBSID>.sap` by removing the pound sign (#) from the corresponding `cleanup_*` entry and specifying another cleanup period:

```
...  
# retention period in days for brbackup log files  
# default: 30  
# cleanup_brbackup_log = 30  
  
# retention period in days for brconnect log files  
# default: 30  
# cleanup_brconnect_log = 30  
...
```

Checking Database Growth

The checks performed by the database system check only give a snapshot of the current situation when BRCONNECT was running. A database administrator must know how fast a specific tablespace has grown to be able to plan new disks. The statistical data about the database growth can be displayed using the menu item *Space* in the DBA Cockpit (transaction DB02).



Caution: This new monitor must then also be activated for the local system. To activate the monitor in the system, perform the following actions (see SAP Note 868063):

1. Start the report RSORACUP
2. Select the following input parameters for the local system:

CON_NAME = DEFAULT

OPERATION = CREATE

or the following for a remote system or external Oracle database:

CON_NAME =<Name of the database connection>

OPERATION = CREATE

Note that the selected database connection according to SAP Notes 323151 and 339092 must already have been imported.

3. Execute (F8)

The new data is displayed after the new collector has successfully run (normally the next day).

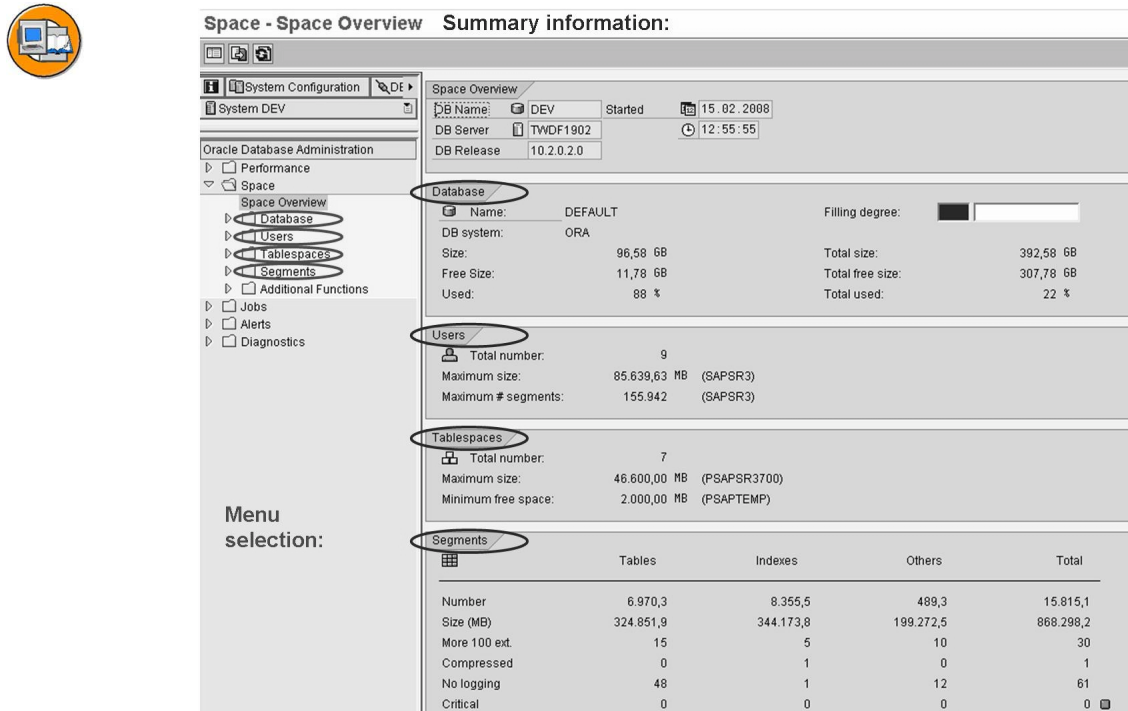


Figure 107: Checking Database Growth

The above figure shows the menu structure and memory space in the DBA Cockpit (transaction DB02). Transaction DB02 provides functions that help analyze the database performance. A historical overview is also available to analyze performance.

Overview of functions in the memory space analysis (menu *Space*) in the DBA Cockpit:

Space Overview

Information overview of the database size and free space.

Database

Overview

Shows memory space statistics of the entire database. To view the total database growth per week or per month, choose the *History* tab.

Users

Overview

Displays information about the database users and memory object that are assigned to them.

Detailed analysis

Displays additional information about database users such as user account status, assigned default tablespaces and whether the user was blocked.

Tablespaces

Overview

Shows the current sizes and free space per tablespace. In addition, tablespace information such as status (online, offline) and backup mode are shown. When a tablespace has autoextensible data files, the total size to which this tablespace can grow by automatically extending all autoextensible data files is shown. In addition, detailed information about data files, such as the current size and the settings of the automatic enhancement is displayed. Shows detailed information about free space in tablespaces, including fragments (free areas within the tablespace). *Free Space Analysis*.

The database views *DBA_DATA_FILES* and *DBA_TEMP_FILES* are displayed.

Detailed analysis

Displays additional detailed information about the individual tablespaces.

Segments

Overview

Displays information such as overall size and number of all segment types in the database. Also displays the largest segments, segments that have the largest allocated number of extents, and segments with the largest growth.

All segments are listed whose next-extent size is larger than the largest free space in the tablespace (Space Critical Objects). You can tell if this is a critical situation:

- If the corresponding tablespace has at least one autoextensible data file **and** one of the data files can be automatically extended to a size that the next extent fits into the data file, the situation is considered a **warning**. As long as there is enough disk space available for any autoextensible data file to grow, no action has to be taken.
- If the corresponding tablespace does not have autoextensible data files, or no autoextensible data file can grow by the next extent size (because MAXBYTES is reached or the disk is full), the situation is **critical**, because adding a single row to this table could cause a tablespace overflow. In this case, either the file system needs to be enlarged (if possible), or the tablespace must be enlarged by adding another data file.

Detailed analysis

Displays additional detailed information about the individual tablespaces.

Detailed analysis (aggregated)

Displays compressed information about selected segments including assigned tables and indexes, partitions, and LOBs.

Additional Functions

Collector Logs

Displays information about the job that collects the data.

BW Analysis

Allows you to display SAP Business Information Warehouse (BW) objects that are assigned to the database.

To collect several statistics about the database, the operating system, and the SAP system, and store the results in database tables for calculating statistics over a certain period of time, the background job *COLLECTOR_FOR_PERFORMANCE_MONITOR* is scheduled hourly by executing the ABAP report RSCOLL00.



Job	Ln	Job Created	Status	Start date
SAP_COLLECTOR_FOR_PERFMONITOR		DDIC	Finished	07.03.2008
SAP_COLLECTOR_FOR_PERFMONITOR		DDIC	Finished	07.03.2008
SAP_COLLECTOR_FOR_PERFMONITOR		DDIC	Finished	07.03.2008
SAP_COLLECTOR_FOR_PERFMONITOR		DDIC	Finished	07.03.2008
SAP_COLLECTOR_FOR_PERFMONITOR		DDIC	Finished	07.03.2008

Background job
COLLECTOR_FOR_PERFORMANCE_MONITOR,
scheduled hourly ...

No.	Program name/command	Prog. type	Spool list	Parameters	User	Lang.
1	RSCOLL00	ABAP			DDIC	EN

... runs report RSCOLL00 ...

... which reads table TCOLL and runs
other check reports on the day and
hour defined here.

ReportName	D	T	D	T	Fri	Sat	D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
RSAMON40	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSCUECRM			X											X																	
RSDBPREV	X	X	X	X	X	X	X	X		X	X			X		X	X	X	X		X		X	X	X	X	X	X	X	X	X
RSDB_DAILY	X	X	X	X	X	X	X	X		X																					
RSDB_HRLY	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSDB_PAR	X	X	X	X	X	X	X									X					X						X				
RSDB_TDB	X	X	X	X	X	X	X								X													X			
RSDB_WDB						X														X											
RSHOSTDB	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSHOSTDC	X	X	X	X	X	X	X	X		X		X		X		X		X		X		X		X		X		X		X	X
RSHOSTPH	X	X	X	X	X	X	X										X				X						X				
RSICFDTL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSICFDMN	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSICFJOB	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSORA811	X	X	X	X	X	X	X													X							X				
RSORACOL	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RSORAHCL								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Collected statistics are written into database tables (MONI and PAHI).

Figure 108: Collecting Regular Statistics

Report RSCOLL00 reads table *TCOLL*, which contains a list of other reports and a matrix specifying the day of the week and hour of the day when this report is to be executed.

These reports collect their data and write information into tables *MONI* (statistical data) and *PAHI* (parameters of operating system, database, and SAP).



Note: For more information about the reports in table *TCOLL*, as well as necessary customizing of *TCOLL* when several instances of an SAP system are running on one host, see SAP Note 12103.

Exercise 10: Database System Check

Exercise Objectives

After completing this exercise, you will be able to:

- Use BRCONNECT to check the database and adapt the NEXT extent size
- Use BR*Tools and SAP transactions to view the results of the checks

Business Example

You want to perform regular database checks.

Task:

In this exercise, a new tablespace is created and filled using SQL scripts. Using BRCONNECT, you will detect and describe any problems.

For this exercise, connect to your database.

1. Change to the subdirectory `G:\oracle\<SID>\scripts` and execute the command file `tscreate.bat`. The command file will create the tablespace `PSAP<SID>EX` and fill it with data.
2. Run the database check. What problems exist for the new tablespace?
3. Run `brconnect -f next` and check how it solves the problems.
4. Check the database again. Did the action of `brconnect -f next` permanently solve the problem?
5. Insert further rows into table *SKEL* by executing the command file `tsfill.bat`. Then check the database again to see if you get a warning. If you execute `tsfill.bat` a second time, you can also see the Oracle error message indicating the tablespace overflow.

Solution 10: Database System Check

Task:

In this exercise, a new tablespace is created and filled using SQL scripts. Using BRCONNECT, you will detect and describe any problems.

For this exercise, connect to your database.

1. Change to the subdirectory G:\oracle\<SID>\scripts and execute the command file tscreate.bat. The command file will create the tablespace PSAP<SID>EX and fill it with data.

a)

```
G:\oracle\T99>cd scripts

G:\oracle\T99\scripts>tscreate.bat
Creating the tablespace PSAPT99EX

G:\oracle\T99\scripts>
```

2. Run the database check. What problems exist for the new tablespace?

- a) BRCONNECT detected that the largest free area within the data file is smaller than the next extent of the table *SKEL*. This is a part of the screen output:

```
G:\oracle\T00>brconnect -c -f check
...
BR0805I Start of BRCONNECT processing: cdwrobht.chk 2007-11-27 09.45.33
BR0484I BRCONNECT log file: G:\oracle\T99\sapcheck\cdwrobht.chk
...
BR0969I Checking database administration...
BR0970W Database administration alert - level: WARNING, type:
CRITICAL_SEGMENT,object: (table) SAPT99.SKEL, value: 504 KB * 1
/ PSAPT99EX (> 480/0/0/0/0 KB)
...
BR0955I Number of signaled error/warning/exception alerts for database
administration: 1/1/0
```

- b) The log file cdkzkdjf.chk indicates that PSAPT99EX is a dictionary-managed tablespace (no + sign at tablespace status) with no autoextensible data file (no + sign at the status of any data file of PSAPT99EX):

```
BR0118I Tablespaces and data files
```

Continued on next page

Tablespace	Status	File			
Status	Id.	Size	MaxSize	IncrSize	BlkSize
Device	Type	Link			
PSAPT99	ONLINE+	G:\ORACLE\T99\SAPDATA3\T99_1\T99.DATA1			
ONLINE+	4	20979712	52428800	2097152	8192
6	FILE	NOLINK			
PSAPT99EX	ONLINE	G:\ORACLE\T99\SAPDATA3\T99EX_1\T99EX.DATA1			
ONLINE	6	2105344	0	0	8192
6	FILE	NOLINK			
PSAPT99USR	ONLINE+	G:\ORACLE\T99\SAPDATA3\T99USR_1\T99USR.DATA1			
ONLINE	5	10493952	0	0	8192
6	FILE	NOLINK			
PSAPTEMP	ONLINE#	G:\ORACLE\T99\SAPDATA2\TEMP_1\TEMP.DATA1			
ONLINE	-1	20979712	0	0	8192
6	FILE	NOLINK			
PSAPUNDO	ONLINE-	G:\ORACLE\T99\SAPDATA2\UNDO_1\UNDO.DATA1			
ONLINE	2	20979712	0	0	8192
6	FILE	NOLINK			
SYSAUX	ONLINE+	G:\ORACLE\T99\SAPDATA1\SYSAUX_1\SYSAUX.DATA1			
ONLINE	3	209723392	0	0	8192
6	FILE	NOLINK			
SYSTEM	ONLINE	G:\ORACLE\T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1			
SYSTEM	1	419438592	0	0	8192
6	FILE	NOLINK			

- c) While the tablespace itself is only 76% full (first *Used[%]* column), the largest free area is 480 KB. The missing + sign in the last *Free[KB]* column indicates again that there is no autoextensible data file:

BR0983I Tablespace fragmentation

Tablespace	Files	Tables	Indexes	Extents	Total [KB]	Used [%]
Free [KB]	FreeExt.	MaxSize [KB]	MaxAlloc [KB]	Used [%]	Used [%]	Free [KB]
Largest [KB]						
PSAPT99	1	26	27	53	20480	16.88
17024	1	51200+		30720+	6.75+	47744+
30720+:17024:0:0:0						
PSAPT99EX	1	1	0	3	2048	76.56
480	1	2048		0	76.56	480
480:0:0:0:0						
PSAPT99USR	1	0	0	0	10240	0.63

Continued on next page

```

10176          1          10240          0          0.63          10176
10176:0:0:0:0
PSAPTEMP          1          0          0          0          20480          0.00
20480          0          20480          0          0.00          20480
0:0:0:0:0
PSAPUNDO          1          0          0          0          20480          0.31
20416          229          20480          0          0.31          20416
1024:1024:1024:1024:1024
SYSAUX          1          309          348          928          204800          29.72
143936          1          204800          0          29.72          143936
143936:0:0:0:0
SYSTEM          1          507          571          2166          409600          44.52
227264          1          409600          0          44.52          227264
227264:0:0:0:0

Total:          7          843          946          3150          688128          36.09
439776          234          718848          30720          34.55          470496
413600:18048:1024:1024:1024

```

3. Run `brconnect -f next` and check how it solves the problems.

a)

```

G:\oracle\T00>brconnect -c -f next
...
BR0907I Checking and adapting next extent of tables and indexes...
BR0917I Next extent size adapted for table SAPR3.SKEL, value old/new:
504 KB / 480 KB

```

- b) `Brconnect -f next` adapts the next extent size to exactly the size of the largest free area within the tablespace. By doing so, the table can grow by another extent.

4. Check the database again. Did the action of `brconnect -f next` permanently solve the problem?

- a) No. The database check still issues a warning. The situation has become a bit better, as a tablespace overflow would occur only when a second next extent would be added: `SAPR3.SKEL, value: 480 KB * 2 / PSAPT00EX (> 480/0/0/0/0 KB)`.

5. Insert further rows into table `SKEL` by executing the command file `tsfill.bat`. Then check the database again to see if you get a warning. If you execute `tsfill.bat` a second time, you can also see the Oracle error message indicating the tablespace overflow.

Continued on next page

- a) Run the command file `tsfill.bat`.

```
G:\oracle\T99\scripts>tsfill.bat
Filling the tablespace PSAPT99EX
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Tue Nov 27 10:06:06 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
Connected.
```

```
2066 rows created.
```

```
2066 rows created.
```

```
2066 rows created.
```

```
Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.2.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options
```

- b) Then check the database again to see if you get a warning.

```
G:\oracle\T00>brconnect -c -f check
```

```
...
```

```
BR0969I Checking database administration...
```

```
BR0970W Database administration alert - level: WARNING, type:
```

```
TABLESPACE_FULL, object: PSAPT99EX, value: 100.00% (> 95%)
```

```
BR0970W Database administration alert - level: WARNING, type:
```

```
CRITICAL_SEGMENT,object: (table) SAPT99.SKEL, value: 480 KB * 1 / PSAPT99EX (> 0/0/0)
```

- c) If you execute `tsfill.bat` a second time, you can also see the Oracle error message indicating the tablespace overflow.

```
G:\oracle\T99\scripts>tsfill.bat
Filling the tablespace PSAPT99EX
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Tue Nov 27 10:14:34 2007
```

```
Copyright (c) 1982, 2005, Oracle. All Rights Reserved.
```

```
Connected.
```

```
INSERT INTO SKEL SELECT * FROM DBA_SEGMENTS
```

```
*
```

Continued on next page

```
ERROR at line 1:  
ORA-01653: unable to extend table SAPT99.SKEL by 60 in tablespace  
PSAPT99EX
```

```
INSERT INTO SKEL  SELECT * FROM DBA_SEGMENTS  
*
```

```
ERROR at line 1:  
ORA-01653: unable to extend table SAPT99.SKEL by 60 in tablespace  
PSAPT99EX
```

```
INSERT INTO SKEL  SELECT * FROM DBA_SEGMENTS  
*
```

```
ERROR at line 1:  
ORA-01653: unable to extend table SAPT99.SKEL by 60 in tablespace  
PSAPT99EX
```

```
Disconnected from Oracle Database 10g Enterprise Edition Release  
10.2.0.2.0 - 64bit Production  
With the Partitioning, OLAP and Data Mining options
```



Lesson Summary

You should now be able to:

- State the purpose of running regular database system checks
- Run regular checks with SAP tools
- Interpret the results of database system checks
- Create a strategy to prevent possible problems and errors

Lesson: CCMS Alert Monitor

Lesson Overview

In this lesson, you will learn to use the CCMS alert monitor to monitor the Oracle database.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain the purpose of the CCMS alert monitor
- Use the CCMS alert monitor to monitor the Oracle database

Business Example

You use the CCMS alert monitor daily to check if there are any warnings or problems within the SAP system. The database is also included in the alert monitor because you want to have a single transaction to view the health of all components in your SAP system.

Introduction to the CCMS Alert Monitor

The alert monitor in CCMS (transaction RZ20) allows you to centrally monitor different parts of the SAP system, including the Oracle database. You can configure analysis and data collection tools for different types of alerts. The database monitor can be found under the monitor set SAP CCMS Monitor Templates.

Monitoring of the Oracle Database

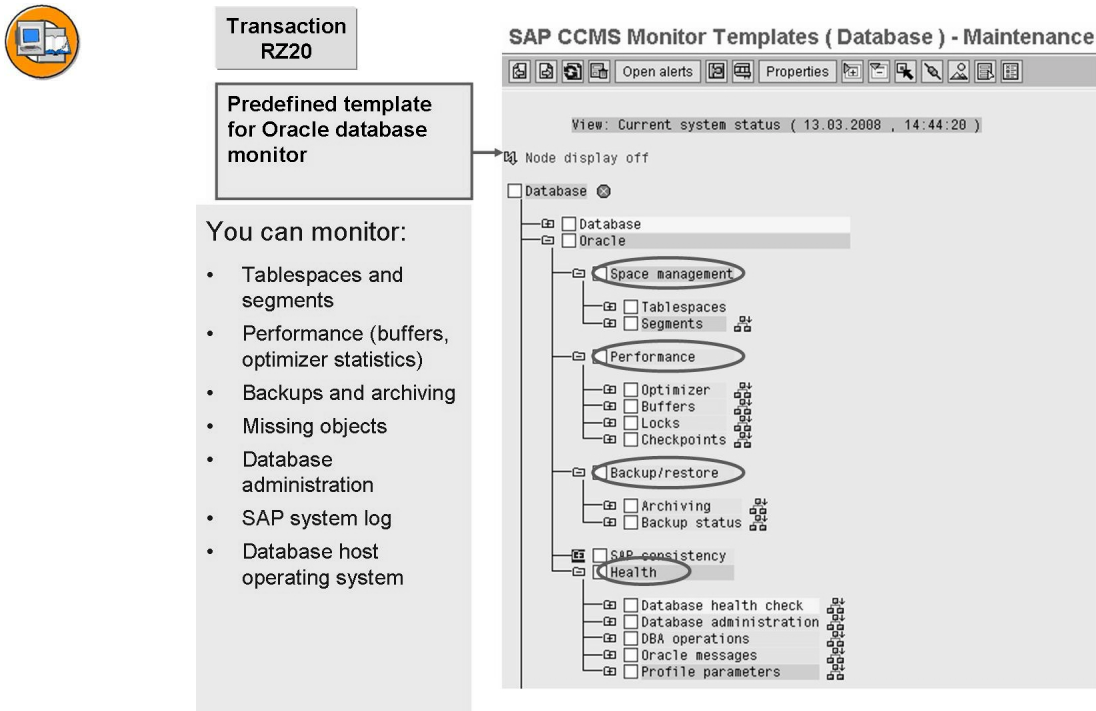


Figure 109: CCMS Alert Monitor

You can use the CCMS alert monitor to check the following Oracle database functions:

- Space management: tablespaces and segments
- Performance: optimizer statistics, buffers, logs, and checkpoints
- Backup or restore: database and redo log backups
- Consistency: between database objects in the ABAP and Oracle Dictionaries
- Health: database system checks from BRCONNECT

In the CCMS monitoring architecture, the monitors get their data from data collectors. The main data collector for the Oracle database functions within the CCMS alert monitor is BRCONNECT. It writes into table *DBMSGORA*, which is then examined by the CCMS alert monitor to display information, warnings, and alerts.



Hint: Whenever you enter new conditions of type ORA (Oracle error messages) or PROF (profile parameter check) in table *DBCHECKORA* using transaction DB17, you must rebuild the monitoring tree to be able to see corresponding alerts in the CCMS alert monitor:

1. Enter the new condition in *DBCHECKORA* using transaction DB17.
2. In transaction RZ20, open *SAP CCMS Monitor Templates*, double-click on the monitor *Database* and activate the maintenance functions by choosing *Extras* → *Activate maintenance function*.
3. Position the cursor on Oracle root MTE (Monitoring Tree Element) and reset all open alerts. Select *Open alerts*, choose *Edit* → *Alerts* → *Reset alerts* and then choose *Reset alerts*.
4. Delete the Oracle monitoring tree by choosing *Edit* → *Delete*
5. Run the ABAP program *RSDBMON0* on the main application server to rebuild the database monitoring tree.

For several other monitoring tree elements within the Oracle part of the CCMS alert monitor, the data collectors directly check V\$- and DBA views from Oracle.



Caution: To avoid the expensive check of view *DBA_SEGMENTS* to display the information in the MTE *Database* → *Oracle* → *Space management* → *Segments*, segment alert information is retrieved from the latest database system check. If this run is older than one day, the data will be retrieved directly from *DBA_SEGMENTS*. This would have a negative effect on the performance. Therefore you need to schedule at least one daily run of `brconnect -f check`.



Lesson Summary

You should now be able to:

- Explain the purpose of the CCMS alert monitor
- Use the CCMS alert monitor to monitor the Oracle database



Unit Summary

You should now be able to:

- Explain how Oracle stores its data
- Explain the difference between dictionary and locally managed tablespaces
- Explain the purpose of undo- and temporary tablespaces
- State the purpose of running regular database system checks
- Run regular checks with SAP tools
- Interpret the results of database system checks
- Create a strategy to prevent possible problems and errors
- Explain the purpose of the CCMS alert monitor
- Use the CCMS alert monitor to monitor the Oracle database



Test Your Knowledge

1. Your database is expected to grow to 300 GB. A good size for data files in this case is _____.
Fill in the blanks to complete the sentence.
2. Oracle stores its data in tablespaces. A tablespace consists of one or more _____. A segment (for example, a table or an index) allocates space within a tablespace using one or more _____.
Fill in the blanks to complete the sentence.
3. For optimal monitoring of the Oracle database within the CCMS alert monitor, you must:
Choose the correct answer(s).
 - ☐ A Maintain check conditions in table DBCHECKORA using transaction DB17.
 - ☐ B Schedule the database system check to run daily.
 - ☐ C Maintain thresholds in the CCMS alert monitor itself.
 - ☐ D Schedule report RSDBMON0 to run daily.



Answers

1. Your database is expected to grow to 300 GB. A good size for data files in this case is 4 GB.

Answer: 4 GB

2. Oracle stores its data in tablespaces. A tablespace consists of one or more data files. A segment (for example, a table or an index) allocates space within a tablespace using one or more extents.

Answer: data files, extents

3. For optimal monitoring of the Oracle database within the CCMS alert monitor, you must:

Answer: A, B, C

Unit 4

Storage Management

Unit Overview

After discussing backup, monitoring, and regular checks on the database, this unit will focus on administrative activities like tablespace administration and reorganization.



Unit Objectives

After completing this unit, you will be able to:

- Create new tablespaces
- Extend existing tablespaces
- Move or rename data files
- Additional database space management options
- Describe a situation in which a reorganization is necessary
- Perform a reorganization
- Transform dictionary-managed tablespaces to locally-managed tablespaces
- List the most common problems that occur in Oracle databases
- Prevent an archiver stuck and solve it when it occurs
- Solve typical problems that occur on productive Oracle databases

Unit Contents

Lesson: Tablespace Administration	296
Exercise 11: Tablespace Administration.....	313
Lesson: Reorganization of Tables	320
Exercise 12: Reorganization of Tables	339
Lesson: Housekeeping and Troubleshooting	343
Exercise 13: Housekeeping and Troubleshooting	357

Lesson: Tablespace Administration

Lesson Overview

In this lesson, you will learn to create and administrate tablespaces.



Lesson Objectives

After completing this lesson, you will be able to:

- Create new tablespaces
- Extend existing tablespaces
- Move or rename data files
- Additional database space management options

Business Example

When monitoring the database, you recognized, that the tablespace PSAP<SID> is more than 98 % full. You check the tablespace statistics: if the tablespace growth is the same as in the last month, the tablespace would be 100 % full in about 4 weeks. As the disks containing tablespaces are already nearly full, you decide to install an additional disk and extend the tablespace by adding a new data file.

Tablespace Administration

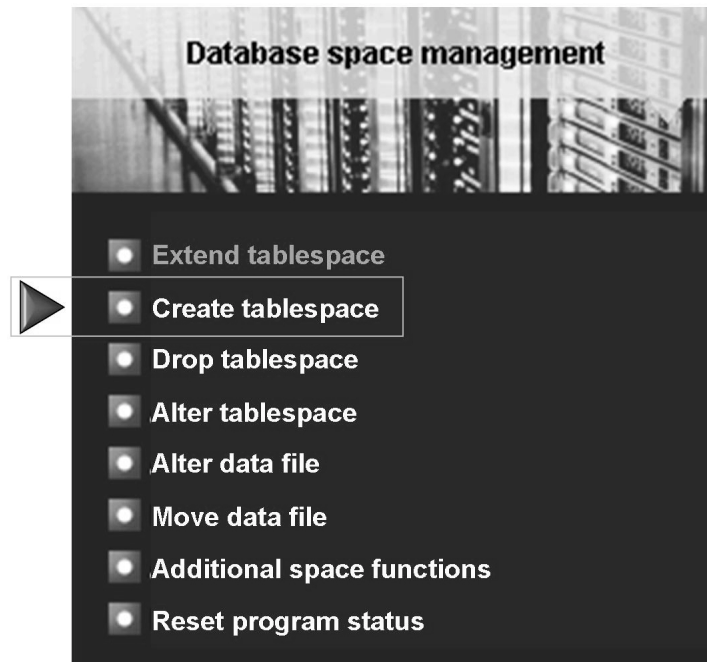


Figure 110: Space management with BR*Tools

The main activities in tablespace administration are:



- Creation of a new tablespace, for example as a preparation to move certain tables out of the standard tablespace PSAP<SCHEMA-ID> into a special tablespace
- Deletion of a tablespace, for example after an upgrade
- Extending tablespaces by adding a data file or tempfile or by resizing existing data files
- Moving data files, for example because after adding a new disk a data file which was created on another disk needs to be moved to the new disk

This lesson describes the actions of BRTOOLS or BRGUI under the main menu item *Space management*.

Creating and Dropping tablespaces

On normal operation creating a new tablespace or dropping an existing tablespace is not necessary.

The creation of a new tablespace is necessary for example

- in preparation for the upgrade: During the upgrade, the old ABAP programs in PSAP<SCHEMA-ID><Old-Rel> are deleted and the new ABAP programs are imported into the new tablespace PSAP<SCHEMA-ID><New-Rel>.
- in preparation for an online reorganization of a tablespace, for example to switch from a dictionary managed to a locally managed tablespace.

To create a new tablespace use BRTOOLS or BRGUI and select *Space management* → *Create tablespace*. To enter the main menu mode of BRSPACE, select *continue* in the next screen and select *Create tablespace* from the BRSPACE menu. As an alternative, you can start `brspace -f tscreate` and press *continue*.. Now fill in the necessary information by selecting the appropriate menu items.



Create a tablespace using BR*Tools

BR0657I Input menu 305 - please check/enter input values

Main options for creation of tablespace in database T99

- 1 - Tablespace name (tablespace) [PSAPT99NEW]
- 2 - Tablespace contents (contents) [data]
- 3 - Segment space management (space) [auto]
- 4 # Database owner of tablespace (owner) . []
- 5 # Table data class / tabart (class) []
- 6 - Data type in tablespace (data) [both]
- 7 # Joined index/table tablespace (join) . []
- 8 ~ Uniform size in MB (uniform) []

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

The main options and their possible entries are:

tablespace

Enter the name of the tablespace to be created. According to the SAP conventions, the tablespace name should start with the letters “PSAP”, followed by the SCHEMA-ID, plus a unique name.

contents

- data for normal tablespaces containing tables and/or indexes
- temp for a temporary tablespace
- undo for an undo tablespace



Hint: A rollback tablespace has the content data, but should only contain rollback segments.

space

You can select either `auto` for automatic segment space management (default) or `manual` for manual segment space management.



Hint: Do not mix automatic/manual segment space management with locally/dictionary managed tablespaces

- automatic segment space management is a new feature of Oracle 9i which offers better performance on parallel queries on a segment
- you cannot create dictionary managed tablespaces with `BRSPACE` due to its disadvantages. `BRSPACE` always creates locally managed tablespaces

class

Option for the table data type when creating a tablespace.

The values `all` | `old_tsp` | `old_tsp_list` trigger data types of all or just specified tablespaces to be transferred during an online reorganization:

- `all` –
transfers all table data types. Is used to reorganize the main tablespace (PSAP<Schema-ID>)
- `old_tsp` | `old_tsp_list` - the new tablespace transfers the table data type(s) from one or several old tablespaces. Is used for the reorganization of one or more tablespaces shifting the tables to a new tablespace

This is relevant when reorganizing tables and simultaneously shifting them to another tablespace (see SAP Note 646681).

The explicitly specified data types `tab_class` | `tab_class_list` must start with the letters "Y" or "Z" (customer data types) and must not be longer than five characters. These data types should be used for customer developments or when shifting individual tables.

If you do not set the option, a customer data type starting with the letter "U" is automatically generated by BRSPACE.

owner

Changing this is only possible if you have multiple components in one database (MCOD). The owner is SAP<SCHEMA-ID> of the database schema you create the tablespace for.

data

In the “old” tablespace layout a tablespace contained either tables (`table`) or indexes (`index`). In the new MCOD tablespace layout, tablespaces contain tables and indexes (`both`). SAP recommends generally to create new tablespaces containing data and indexes to simplify administration.

join

If in the *data* field you entered either `table` or `index`, you can enter here the corresponding index or table tablespace. In the first case, BRSPACE will create both tablespaces. In the second case, BRSPACE creates only a index tablespace on joins it with already existing table tablespace. When you entered `both` in the *data* field, this option is not selectable.

After continuing, the menu for the data file properties is shown. The screenshot shows the output after changing autoextent to yes.



Options for creation of a tablespace

BR0657I Input menu 306 - please check/enter input values

Space options for creation of tablespace PSAPT99NEW (1. file)

```

1 - Tablespace file name (file) ..... [G:\oracle\T99\sapdata3\t99new_1\t99
new.data1]
2 # Raw disk / link target (rawlink) ..... []
3 - File size in MB (size) ..... [2]
4 - File autoextend mode (autoextend) .... [no]
5 # Maximum file size in MB (maxsize) .... []
6 # File increment size in MB (incrsize) . []
7 - SQL command (command) ..... [create tablespace PSAPT99NEW extent
management local autoallocate segment space management auto datafile 'G:\oracle
\T99\sapdata3\t99new_1\t99new.data1' size 2M autoextend off]

```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

The main options and their possible entries are:

file

The name of the new data file for the tablespace. The default path and file name follows the SAP naming conventions for data files. You can create a tablespace with up to five data files using BRSPACE.



Hint: You can use just the number of the sapdata<n> directory as a shortcut for the file name, for example, “4” generates a new tablespace file name which is located in the sapdata4 directory.



Hint: To create a tablespace with more than five data file, first create the tablespace and then extend the tablespace by adding any number of additional data files.

rawlink

Soft link to a raw disk or to an external (non-sapdata) directory if there is no free space available in the sapdata directory.

size

Size of the data file in MB.

autoextend

If set to `yes`, the new data file is created autoextendible. In this case, `maxsize` and `incrsize` must be specified. If set to `no`, `maxsize` and `incrsize` cannot be specified and their entries are locked.

maxsize

For autoextendible data files, this parameter specifies the maximum size in MB up to which the data file can be increased.

incrsize

For autoextendible data files, this parameter specifies the size in MB, by which the data file is automatically increased when necessary.

Now the tablespace will be created. Because creating a tablespace is a structural change in the database, a control file backup is created in the directory `$SAPDATA_HOME/sapreorg/<encoded timestamp>` before and after the creation. In addition the action will be reported in `$SAPDATA_HOME/sapreorg/struc<DBSID>.log`.

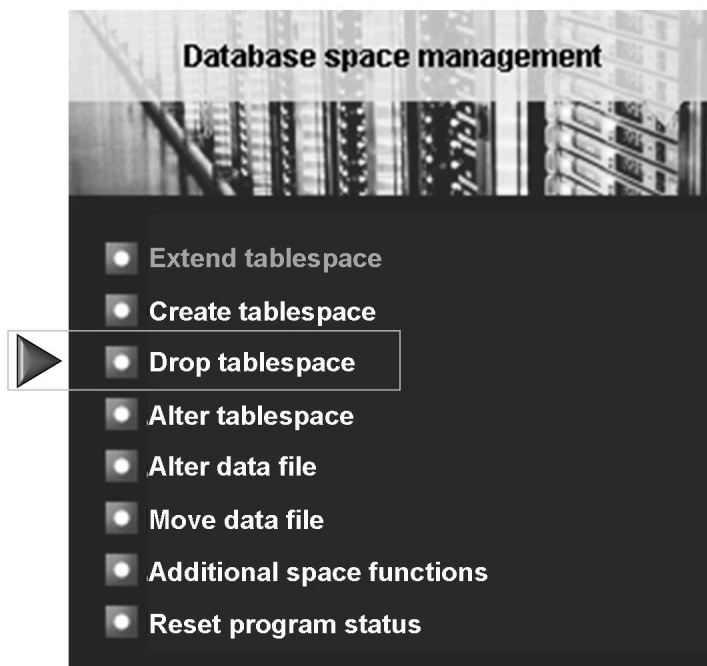


Figure 111: Dropping a tablespace with BR*Tools

To drop a tablespace use BRTOOLS or BRGUI and select *Space management* → *Drop tablespace*. To enter the main menu mode of BRSPACE, select *continue* in the next screen and select *Drop tablespace* from the BRSPACE menu. As an alternative, you can start `brspace -f tsdrop` and enter *continue*.

Now a list of tablespaces is shown, from where you can select the tablespace to be dropped. After selecting the tablespace you can set the force option (default: no). Per default, BRSPACE will not drop a tablespace which is not empty. Using the *force* option, BRSPACE will drop the tablespace even if it is not empty.

When a tablespace is dropped, BRSPACE will

- create a control file backup in the directory
\$SAPDATA_HOME/sapreorg/<encoded timestamp>
before and after the deletion
- check if the tablespace is empty. If the tablespace is not empty, it will not be dropped unless you force it by setting the force mode to yes.
- take the tablespace offline
- drop the tablespace including data files
- remove all subdirectories for the data files
- create entries for the delete process in `struc<DBSID>.log`

Enlarging Tablespaces

To make a tablespace larger, there are three possible ways:

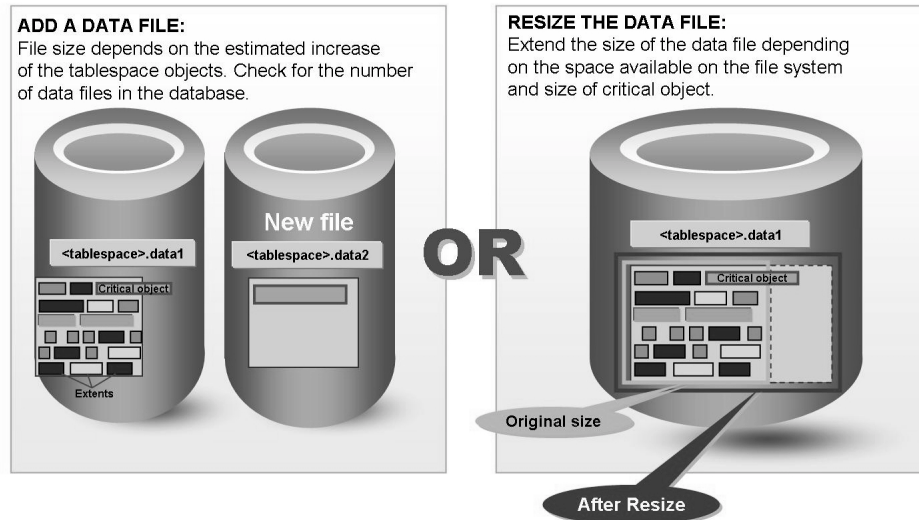


Figure 112: Enlarging a tablespace

- a new data file can be added to the existing tablespace.
 Use this option, if the existing data files cannot be resized because there is no disk space available on the disks holding the existing data files, or because the existing data files already have their maximum size and you don't want to make them larger.
 - the properties of an existing data file can be changed to be autoextensible.
 Use this option to simplify further tablespace administration in the future. Any autoextensible data file will grow automatically when needed up to a maximum file size defined.
- Caution:** When using autoextensible data files, you have to monitor the disk space usage. In this case, a tablespace overflow can still occur if the disks holding these data files are full.
- an existing data file can be resized.
 Use this option if you want to keep the control over the data file growth.
- You can create a new data file under the menu option *Extend tablespace*, and you can find the properties and size of the data file under the menu option *Alter data file*.

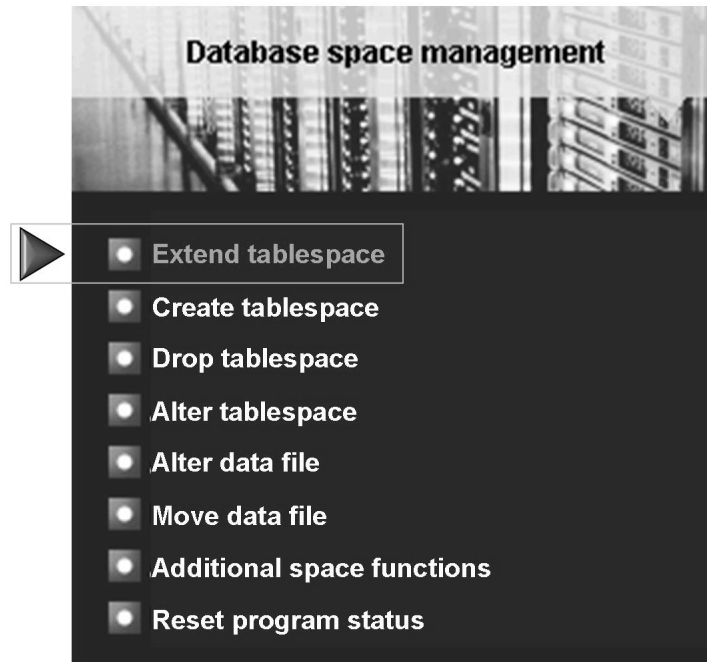


Figure 113: Extending a tablespace with BR*Tools

To enlarge a tablespace by adding a new data file start BRTOOLS or BRGUI and select *Space management* → *Extend tablespace*. To enter the main menu mode of BRSPACE, select *continue* in the next screen and select *Extend tablespace* from the BRSPACE menu. As an alternative, you can use `brspace -f tsextend` and enter *continue*.

After selecting the tablespace to be extended, a menu similar to the menu when creating a tablespace is shown. Enter the appropriate values as described above when creating a tablespace.

Changing the Attributes of Data Files

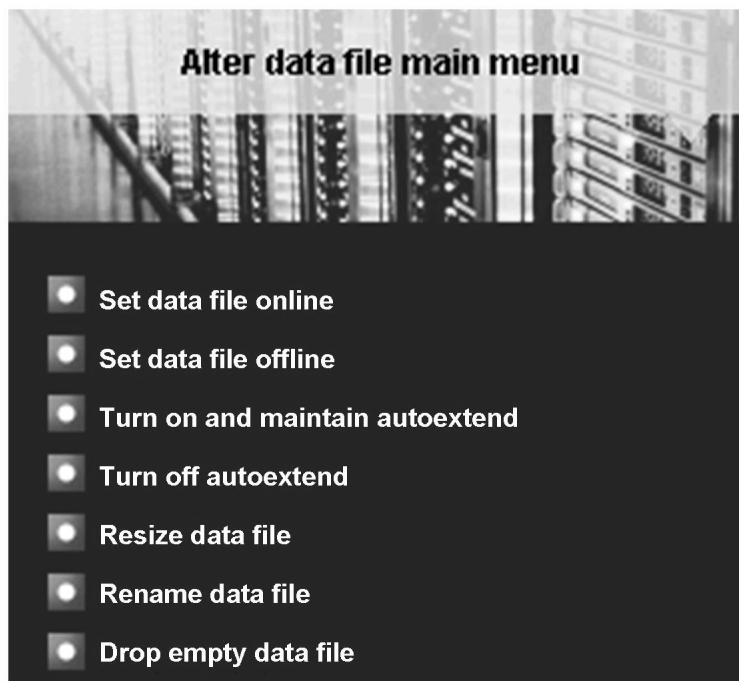


Figure 114: Changing the attributes of data files using BR*Tools

Resize data file

To resize an existing data file start BRTOOLS or BRGUI and select *Space management* → *Alter data file* → *Resize data file*. To enter the main menu mode of BRSPACE, select *continue* in the next screen. As an alternative, you can use `brspace -f dfalter` and select *Resize data file*.

Now select the data file from the list. Specify the new data file size and continue.

Turn on and maintain autoextend

To switch an existing file to be autoextensible start BRTOOLS or BRGUI and select *Space management* → *Alter data file* → *Turn on and maintain autoextend*. To enter the main menu mode of BRSPACE, select *continue* in the next screen. As an alternative, you can use `brspace -f dfalter` and select *Turn on and maintain autoextend*.

Now select the data file from the list. The list only contains all data files because you can change `maxsize` and `incrsize` with this function.

In the next screen, the parameters for automatic extension of the data file have to be entered (`maxsize`, `incrsize` - see above).

Turn off autoextend

To reset an existing file to be autoextensible start BRTOOLS or BRGUI and select *Space management* → *Alter data file* → *Turn off autoextend*.

Rename data file

The existing function for shifting data files is only available when you change the sapdata directory. This means that it is not possible to “shift” and rename data files in a sapdata directory. You can now use the new action *Rename data file* in the BRSPACE function *Alter data file* to do this.



Hint: The new name can also be entered interactively in the menu *Options for alter of data file*.



Caution: The new name for the database file should correspond to the SAP naming conventions.

Drop empty data file

If a data file was created with an invalid size or in the wrong directory, this can usually be corrected using RESIZE or RENAME. If you want to drop the data file again instead, the following options are possible:

- Up to and including Oracle 9i, you can only delete a created data file during a tablespace reorganization. There are no other possible options.
- As of Oracle 10g, an empty data file can also be dropped

BRSPACE now supports this feature using a new action *Drop empty data file* within the function *Alter data file* (see SAP Note 592393).



Hint: If extents still exist in the data file, this command fails with ORA-03262. In this case, the relevant segments must first be stored so that the extents can be released.

Moving or Renaming Data Files

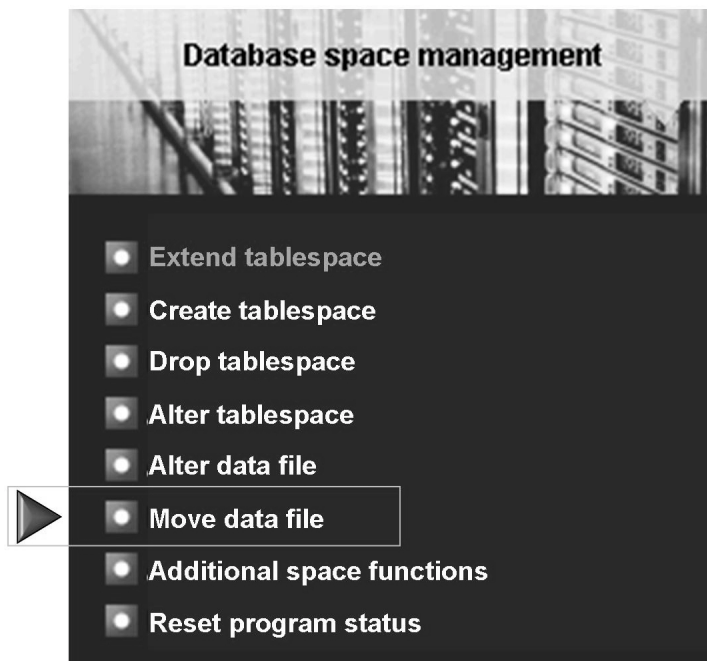


Figure 115: Moving a data file with BR*Tools

To move or rename a data file, it is not sufficient to just move or rename the file itself, as Oracle would not know about the new file name or location. After moving or renaming a data file, the new file name has to be entered in the control file with Oracle commands. Using BRSPACE, all necessary actions are performed.

You might want to move an existing data file to another location if

- you want to move data files from a file system to raw devices or from raw devices to a file system (on UNIX)
- you had to extend a tablespace by adding a new data file, but because of lack of disk space in the disks containing sapdata directories, the data file was created on another disk which was not intended to hold data files. After adding another disk you want to move this data file to the new disk.
- you need to replace a disk on Windows (due to different drive letters used).



Caution: BRSPACE can only move data files which were created according to the SAP conventions. If you want to move a data file created not in a sapdata directory, you have to move the data file with Oracle tools.

If you need to create a data file on a non-standard sap-data disk, therefore you should create the directory path <drive:>\oracle\<SID>\sapdata<n> on Windows before and create the new data file there!

To move or rename a data file start BRTOOLS or BRGUI and select *Space management* → *Move data file*. To enter the menu mode of BRSPACE, select *continue* in the next screen. As an alternative, you can use `brspace -f dfmove` and enter *continue*. Now select the data file to move from the list of data files.



Hint: You can select one or more data files, separated by a comma, or a range of data files separated by a minus sign.

In the next screen, enter the required information:

destination

enter the full path of the sapdata directory where data file shall be moved to. This directory must already exist. The subdirectory for the data file must not be entered and will be created by BRSPACE.

parallel

when moving more than one data file, you can parallelize the copy process. Enter the number of copy processes here.

force

To move or rename a data file, the database needs to be shut down. BRSPACE will normally only shut down the database, if SAP is not running. You can force BRSPACE to shut down the database even when SAP is connected by setting the force option.

Additional Database Space Management Options

To show database information start BRTOOLS or BRGUI and select *Space management* → *Additional space functions*.

You can view information about tablespace, data files, redo log and control files. In addition, you can display information about the disk usage of all directories with database data.

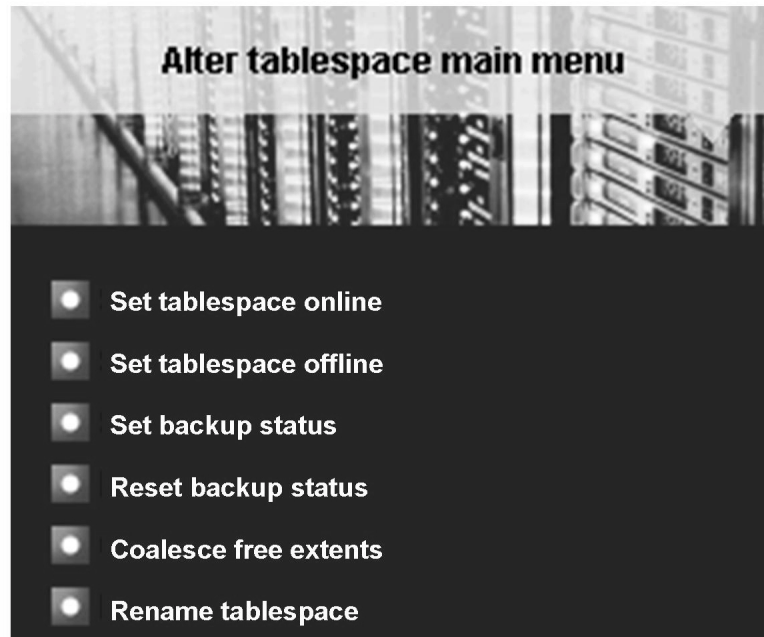


Figure 116: Alter tablespace with BR*Tools

To alter a tablespace start BRTOOLS or BRGUI and select *Space management* → *Alter tablespace*. Now you have the following possibilities:

Set tablespaces offline and online

When a tablespace is set offline, it cannot be accessed anymore. For SAP to work correctly, all tablespaces belonging to the corresponding database schema user SAP<SCHEMA-ID> must be online!

There are situations, when Oracle sets a tablespace offline. This happens for example when Oracle receives an I/O error from the operating system when writing into a data file. To avoid corrupt blocks to be written, Oracle sets this tablespace offline.



Caution: Whenever a tablespace is offline, and it was not actively set offline by a database administrator, you must find the reason for this and solve the problem causing Oracle to set this tablespace offline! After finding and resolving the problem, the tablespace can be set online using this menu.

In databases having multiple components in one database (MCOD), the tablespaces belongs to one schema user SAP<SCHEMA-ID> could be set offline to perform certain administrative actions without affecting other SAP systems using a different SAP<SCHEMA-ID> user. But even in this situation, the tablespaces *SYSTEM*, *PSAPROLL/PSAPUNDO* and *PSAPTEMP* must not be set offline, because these tablespaces are used by all schema users.

Set / Reset backup status

The backup status is regularly set by BRBACKUP when performing an online backup and reset after the backup. When BRBACKUP crashed during an online backup, one or more tablespaces remain in backup mode. This situation is checked by the database system check, which reports the following warning if a tablespace is in backup mode: BR0970W Database administration alert - level: WARNING, type: TABLESPACE_IN_BACKUP, object: PSAPT99

In this case,

- a normal shutdown would not work
- if the database crashes or a shutdown abort is performed while a tablespace is in backup mode, the database would need a manual recovery in order to be opened.

To avoid a manual recovery, reset the backup status when a tablespace is in backup mode when the following conditions are true:

- You notice that an online backup has crashed
- There is currently no online backup running.

Coalesce free extents

Coalescing free extents means that several free extents which directly follow each other are combined to a single, larger free extent. The tool database system check will automatically coalesce free extents for all tablespaces. Manually coalescing free extents using this menu makes sense when a high amount of data was deleted in the database, for example after archiving data or the deletion of a client.

Rename Tablespace

As of Oracle 10g, the name of tablespaces can be changed. BRSPACE supports this feature using a new action *Rename tablespace* in the function *Alter tablespace*.

The data files are also renamed. This action requires that the new tablespace is set to “offline” in the short term, which can affect the SAP system. It can also be set to run at a specific time when the system has a minimum load (see SAP Note 914174).

Exercise 11: Tablespace Administration

Exercise Objectives

After completing this exercise, you will be able to:

- Create tablespaces
- Enlarge tablespaces

Business Example

The database system check shows that a tablespace is nearly full. You want to enlarge the tablespace to provide additional space.

Task:

Tablespace *PSAP<DBSID>EX* needs to be enlarged to provide additional space. Furthermore, a new locally managed tablespace needs to be created.

1. Extend tablespace *PSAP<DBSID>EX* (created in the previous exercise) by resizing its data file to 4 MB. Use the database system check to see if, after this action, enough free space is available.
2. Set the data file of tablespace *PSAP<DBSID>EX* to autoextensible. The maximum size should be 10 MB, the increment size 2 MB. Check the size of the data file, then run script `tsfill.bat`. Check the size of the data file again; it should have grown by 2 MB.
3. Create a new locally managed tablespace, *PSAP<DBSID>NEW*, with one autoextensible data file with a size of 4 MB, increment size 2 MB, and maximum size of 10 MB.

Solution 11: Tablespace Administration

Task:

Tablespace *PSAP<DBSID>EX* needs to be enlarged to provide additional space. Furthermore, a new locally managed tablespace needs to be created.

1. Extend tablespace *PSAP<DBSID>EX* (created in the previous exercise) by resizing its data file to 4 MB. Use the database system check to see if, after this action, enough free space is available.

- a) Start BRGUI or BRTOOLS and choose *Space management → Alter data file → Resize data file* . Select "Continue" until the list of data files is displayed:

```
BR0659I List menu 315 + please select one or more entries
```

```
-----
List of data files for alter
```

Pos.	Tablespace	Status	Type	Size[KB]	AuExt.	File
1 -	PSAPT99	ONLINE	FILE	20480	YES	G:\ORACLE\T99\SAPDATA3\T99_1\T99.DATA1
2 -	PSAPT99EX	ONLINE	FILE	2048	NO	G:\ORACLE\T99\SAPDATA3\T99EX_1\T99EX.DATA1
3 -	PSAPT99USR	ONLINE	FILE	10240	NO	G:\ORACLE\T99\SAPDATA3\T99USR_1\T99USR.DATA1
4 -	PSAPTEMP	ONLINE	FILE	20480	NO	G:\ORACLE\T99\SAPDATA2\TEMP_1\TEMP.DATA1
5 -	PSAPUNDO	ONLINE	FILE	20480	NO	G:\ORACLE\T99\SAPDATA2\UNDO_1\UNDO.DATA1
6 -	SYS_AUX	ONLINE	FILE	204800	NO	G:\ORACLE\T99\SAPDATA1\SYS_AUX_1\SYS_AUX.DATA1
7 -	SYSTEM	SYSTEM	FILE	409600	NO	G:\ORACLE\T99\SAPDATA1\SYSTEM_1\SYSTEM.DATA1

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
```

```
-----
BR0662I Enter your selection:
```

- b) Select the data file *PSAP<DBSID>EX* and enter **4** as the new data file size in MB. The database system check should not display any warning or error regarding free space in any tablespace.

Continued on next page

When running the script `tsfill.bat` several times , a tablespace overflow will occur again:

```
G:\oracle\T00\scripts>tsfill
...
...
...
G:\oracle\T99\scripts>tsfill.bat
Filling the tablespace PSAPT99EX

SQL*Plus: Release 10.2.0.2.0 - Production on Tue Nov 27 10:34:10 2007

Copyright (c) 1982, 2005, Oracle. All Rights Reserved.

Connected.
INSERT INTO SKEL SELECT * FROM DBA_SEGMENTS
*
ERROR at line 1:
ORA-01653: unable to extend table SAPT99.SKEL by 60 in tablespace
PSAPT99EX

INSERT INTO SKEL SELECT * FROM DBA_SEGMENTS
*
ERROR at line 1:
ORA-01653: unable to extend table SAPT99.SKEL by 60 in tablespace
PSAPT99EX

INSERT INTO SKEL SELECT * FROM DBA_SEGMENTS
*
ERROR at line 1:
ORA-01653: unable to extend table SAPT99.SKEL by 60 in tablespace
PSAPT99EX
```

2. Set the data file of tablespace *PSAP<DBSID>EX* to autoextensible. The maximum size should be 10 MB, the increment size 2 MB. Check the size of the data file, then run script `tsfill.bat`. Check the size of the data file again; it should have grown by 2 MB.

Continued on next page

- a) Start BRGUI or BRTOOLS and choose *Space management* → *Alter data file* → *Turn on and maintain autoextend*. Continue until the list of data files is displayed. Select the data file *PSAP<DBSID>EX* and, in the next menu, set the file increment size to **2 MB** and the maximum file size to **10 MB**:

```
BR0657I Input menu 316 - please check/enter input values
-----
Options for alter of data file G:\ORACLE\T99\SAPDATA3\T99EX_1\T99EX
.DATA1

1 * Current data file status (status) ..... [FIXSIZE]
2 * Current data file size in MB (currsize) . [4]
3 * Alter data file action (action) ..... [autoext]
4 - Maximum file size in MB (maxsize) ..... [10]
5 - File increment size in MB (incrsize) .... [2]
6 # New data file size in MB (size) ..... [4]
7 # New data file name (name) ..... []
8 # Force data file alter (force) ..... [no]
9 - SQL command (command) ..... [alter database
    datafile 'G:\ORACLE\T99\SAPDATA3\T99EX_1\T99EX.DATA1'
    autoextend on next 2M maxsize 10M]

Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
BR0662I Enter your choice:
```

- b) Check the size of the data file before and after running *tsfill.bat*:

```
G:\oracle\T00>cd scripts

G:\oracle\T99\scripts>dir G:\Oracle\T99\sapdata3\T99EX_1
Volume in drive G is Database
Volume Serial Number is BA5E-C3C9

Directory of G:\Oracle\T99\sapdata3\T99EX_1

11/27/2007  09:42 AM      <DIR>          .
11/27/2007  09:42 AM      <DIR>          ..
11/27/2007  10:36 AM              4,202,496 T99EX.DATA1
               1 File(s)          4,202,496 bytes
               2 Dir(s)  821,317,136,384 bytes free
```

Continued on next page

```

G:\oracle\T99\scripts>tsfill.bat
Filling the tablespace PSAPT99EX

SQL*Plus: Release 10.2.0.2.0 - Production on Tue Nov 27 10:41:28 2007

Copyright (c) 1982, 2005, Oracle. All Rights Reserved.

Connected.

2066 rows created.

2066 rows created.

2066 rows created.

Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.2.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options

G:\oracle\T99\scripts>dir G:\Oracle\T99\sapdata3\T99EX_1
Volume in drive G is Database
Volume Serial Number is BA5E-C3C9

Directory of G:\Oracle\T99\sapdata3\T99EX_1

11/27/2007  09:42 AM      <DIR>          .
11/27/2007  09:42 AM      <DIR>          ..
11/27/2007  10:41 AM             6,299,648 T99EX.DAT1
               1 File(s)             6,299,648 bytes
               2 Dir(s)  821,312,806,912 bytes free

G:\oracle\T99\scripts>

```

3. Create a new locally managed tablespace, *PSAP<DBSID>NEW*, with one autoextensible data file with a size of 4 MB, increment size 2 MB, and maximum size of 10 MB.
 - a) Start BRGUI or BRTOOLS and chooset *Space management* → *Create tablespace*. In the BRSPACE menu Main options for creation of tablespace in database <DBSID>, enter **PSAP<DBSID>NEW** as the new tablespace name:

Continued on next page

```

BR0657I Input menu 306 - please check/enter input values
-----
Space options for creation of tablespace PSAPT99NEW (1. file)

1 - Tablespace file name (file) ..... [G:\oracle\T99\sapdata3\
    \t99new_1\t99new.data1]
2 # Raw disk / link target (rawlink) ..... []
3 - File size in MB (size) ..... [6]
4 - File autoextend mode (autoextend) .... [yes]
5 - Maximum file size in MB (maxsize) .... [10]
6 - File increment size in MB (incrsize) . [2]
7 - SQL command (command) ..... [create tablespace
    PSAPT99NEW extent management local autoallocate segment
    space management auto datafile 'G:\oracle\T99\sapdata3\
    t99new_1\t99new.data1' size 6M autoextend on next 2M
    maxsize 10M]

Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
BR0662I Enter your choice:

```

b) Define the file parameters.

```

Space options for creation of tablespace PSAPT99NEW (1. file)

1 - Tablespace file name (file) ..... [G:\oracle\T99\sapdata3\
    t99new_1\t99new.data1]
2 # Raw disk / link target (rawlink) ..... []
3 - File size in MB (size) ..... [4]
4 - File autoextend mode (autoextend) .... [yes]
5 - Maximum file size in MB (maxsize) .... [10]
6 - File increment size in MB (incrsize) . [2]
7 - SQL command (command) ..... [create tablespace
    PSAPT99NEW extent management local autoallocate segment
    space management auto datafile 'G:\oracle\T99\sapdata3\
    t99new_1\t99new.data1' size 4M autoextend on next 2M
    maxsize 10M]

```

Then choose "continue".



Lesson Summary

You should now be able to:

- Create new tablespaces
- Extend existing tablespaces
- Move or rename data files
- Additional database space management options

Lesson: Reorganization of Tables

Lesson Overview

In this lesson you will learn how to perform an online reorganization of tables, indexes, and tablespaces.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe a situation in which a reorganization is necessary
- Perform a reorganization
- Transform dictionary-managed tablespaces to locally-managed tablespaces

Business Example

As your SAP system was installed years ago, you still have dictionary-managed tablespaces. Now you have learnt the advantages of locally managed tablespaces, you want to transform your dictionary managed tablespaces to locally managed tablespaces.

Introduction

Reorganization is generally performed to defragment database objects. By doing so, you can improve performance and regain space within tablespaces that was being unused due to fragmentation.

The classic method of carrying out a reorganization is:



- Export the tables being reorganized (single tables or all tables of a tablespace)
- Drop the exported tables (including the corresponding tablespace if all tables were exported)
- Recreate the tablespace if it was dropped before
- Import the tables that were exported before

This procedure (and other improved, but currently restricted procedures) created objects in the tablespaces contiguously and continuously. This improved performance and removed fragmentation.

On the other hand, reorganization had to be performed offline and took a considerable amount of time when large tables or whole tablespaces had to be reorganized.

With this in mind, the first question is usually: Is reorganization really necessary?

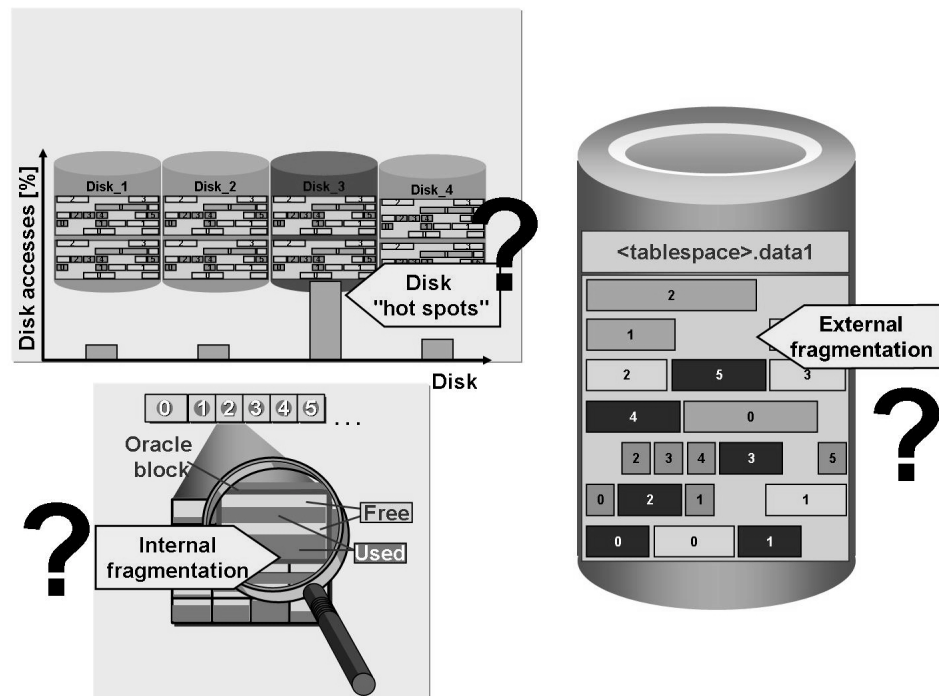


Figure 117: Reasons for Reorganization

Starting with Oracle 8 and now with Oracle 9i, but also due to evolving disk and RAID technology, new features were introduced to make a reorganization less necessary than before:

- Using locally managed tablespaces, space allocation within a tablespace has become more efficient. There is no MAXEXTENTS parameter anymore to stop Oracle from allocating new extents. There is also no NEXT parameter, which could cause Oracle to create too many or too large extents if set incorrectly.
- Using automatic segment space allocation, the chance of internal fragmentation within Oracle blocks is significantly reduced and the performance of parallel queries is improved.
- Having large disks and RAID systems with large and secure memory buffers, I/O hot spots are significantly reduced. Spreading database I/Os to different controllers and disks is performed by a good setup of the RAID system instead of distributing data files manually on different disks.

Using new Oracle online redefinition facilities, reorganization (or better online table redefinition) has become much easier. Advantages and features are that online table redefinition:

- Can be performed online
- Can be parallelized for faster reorganization, or not parallelized to not disturb productive use of the database
- Can be used to move tables to a different tablespace in the same schema
- Can be used to re-create a table to reduce fragmentation
- Introduces less risk, as checks are performed before the redefinition and objects being reorganized are only deleted after the new reorganized version of the object has been successfully created



Hint: The reorganization functionality of BRSPACE is only available for databases as of Oracle 9i. To perform reorganization on previous Oracle versions, you must use SAPDBA. To transform dictionary-managed tablespace to locally managed tablespaces on Oracle 8.1.7, see SAP Note 214995.

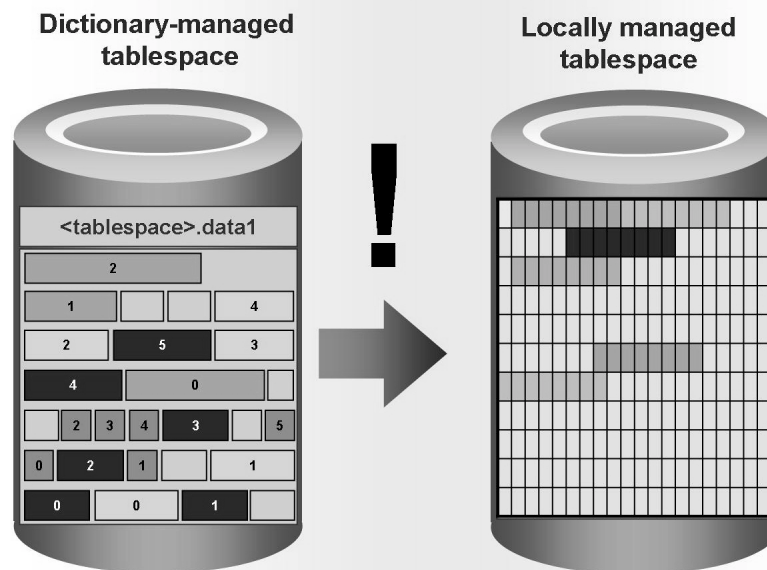


Figure 118: Reasons for Reorganization (2)

Possible reasons for a reorganization include:

- You have fragmented tables or fragmented or degenerated indexes.
- You want to transform dictionary-managed tablespaces to locally-managed tablespaces
- You want to move certain large and heavily used tables into separate tablespaces.

The tool BRSPACE supports online table redefinition as well as traditional reorganization methods. To perform a reorganization, start BRTOOLS or BRGUI and choose *Segment management*. The following list gives an short introduction of the different reorganization methods offered by the menu *Segment management*, and for which kind of reorganization they are used. The next section introduces the reorganization methods in detail.

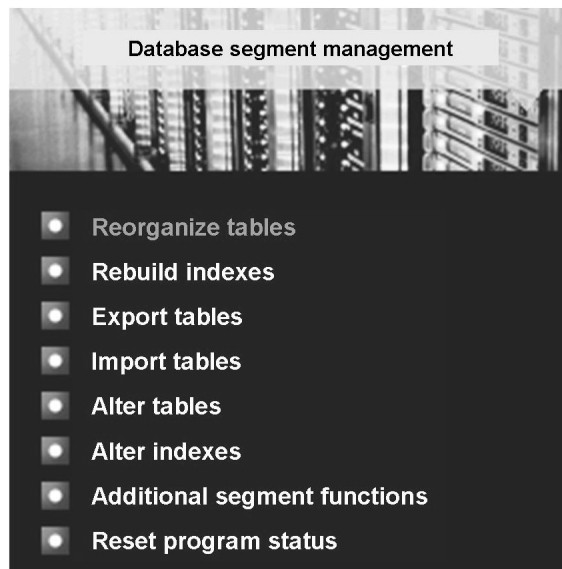


Figure 119: Reorganization with BR*Tools

Reorganize tables

Using menu *Reorganize tables*, BRSPACE will perform an online redefinition of tables. Use this function to:

- Reorganize tables due to internal or external fragmentation,
- Transform dictionary-managed tablespaces to locally-managed tablespaces,
- Transform tablespaces of traditional layout (different tablespaces for data and indexes) into tablespaces of MCOD layout, and vice versa,
- Move large tables to a separate tablespace.

Rebuild indexes

Use this function to rebuild fragmented or degenerated indexes. This will normally improve performance of the table's data access through indexes.

Export tables and Import tables

Use these functions to reorganize tables using the old reorganization method by exporting tables, dropping them, and importing them again.

Normally, tables are reorganized using the BRSPACE function *Reorganize tables*, which will perform an online redefinition of tables. But online redefinition is not possible in Oracle 9i for tables containing LONG fields (existing, for example, in SAP table clusters).



Hint: Oracle has announced that the support of LONG fields will be discontinued as of Oracle 11. As of Oracle 10g, online redefinition will be possible for tables containing LONG fields; LONG fields will then be redefined as LOB (Large Object) fields. Consequently, reorganization tables containing LONG fields in Oracle 9i must be performed by export/import. SAP will support accessing LOB fields through LONG interface as of Oracle 10g, then it will be possible to redefine tables containing LONG fields into tables with LOB fields using BRSPACE. SAP plans to support conversion from LONG to LOB fields. Then conversion can be done online with BRSPACE (Oracle 10g). After the conversion, all SAP tables can be reorganized online.

Alter tables and Alter indexes

The functions for altering tables or indexes actually do not reorganize a table or index, but turn certain features affecting performance on or off.

- *Switch on table monitoring:* With table monitoring switched on, Oracle will automatically collect statistics for these tables. This feature will significantly improve the creation of statistics. When checking and updating the statistics, BRCONNECT must first determine if the number of rows in a table has changed since the last creation of statistics. For tables with monitoring switched off, the change of the number of rows in tables must be determined through collecting statistics on unique indexes. For tables having monitoring switched on, BRCONNECT can easily select information about table changes from *DBA_TAB_MODIFICATIONS* without updating index statistics.



Hint: As of Oracle 9i, It is generally recommended to switch table monitoring on for all SAP tables. In this case, you can schedule a daily statistics update. This means that the statistics are more up-to-date because the runtime only takes up a fraction of the update time without table monitoring.

- *Set parallel degree* is possible for tables and indexes and can improve performance on INSERT statements.



Caution: Only set parallel degree if you are told to do so by SAP support. On productive systems, normally no parallelization is used.

General Steps for Online Table Redefinition

When performing an online table redefinition, BRSPACE uses the PL/SQL package `DBMS_REDEFINITION`, provided by Oracle. Redefinition is done in several steps (see the figure: Online Redefinition Process):

1. Tables are checked to see if they can be redefined online. If the Oracle package returns an error on this check for one or more tables, BRSPACE will exclude them from online redefinition (for example, tables having LONG fields).
2. An empty interim table is created. To create the DDL statement for this step, BRSPACE uses PL/SQL package `DBMS_METADATA`. BRSPACE names the interim table `<SOURCE_TABLE>#`.
3. The redefinition process is started, which will copy data from the original table into the interim table. You can run this as a parallel process with the option `-e | -degree`.

Calling `DBMS_REDEFINITION.START_REDEF_TABLE`:

- Wait until all open changes have been committed to the table to be reorganized.
 - Copy all data from the source table to the target table.
 - Logging all changes to the source table in a Materialized View Log (`MLOG$_<SOURCE_TABLE>`)
4. Create all indexes, constraints, triggers, grants, and comments in the interim table. Referential constraints are disabled. BRSPACE will name the triggers, indexes, and constraints `<ORIGINAL_NAME>#`.
 5. The redefinition is finished using a procedure from the `DBMS_REDEFINITION` package. When this process ends, the original table is redefined, and refers to all attributes, indexes, constraints, grants, and triggers in the interim table. Indexes, triggers, grants, constraints, and comments from the original table are transferred to the interim table. Referential constraints are enabled.
 6. Objects created in step 4 are now renamed by BRSPACE to their original name.
 7. As the last step, BRSPACE will drop the interim table, causing indexes, constraints, triggers, grants, and comments defined on the original table to be dropped as well.

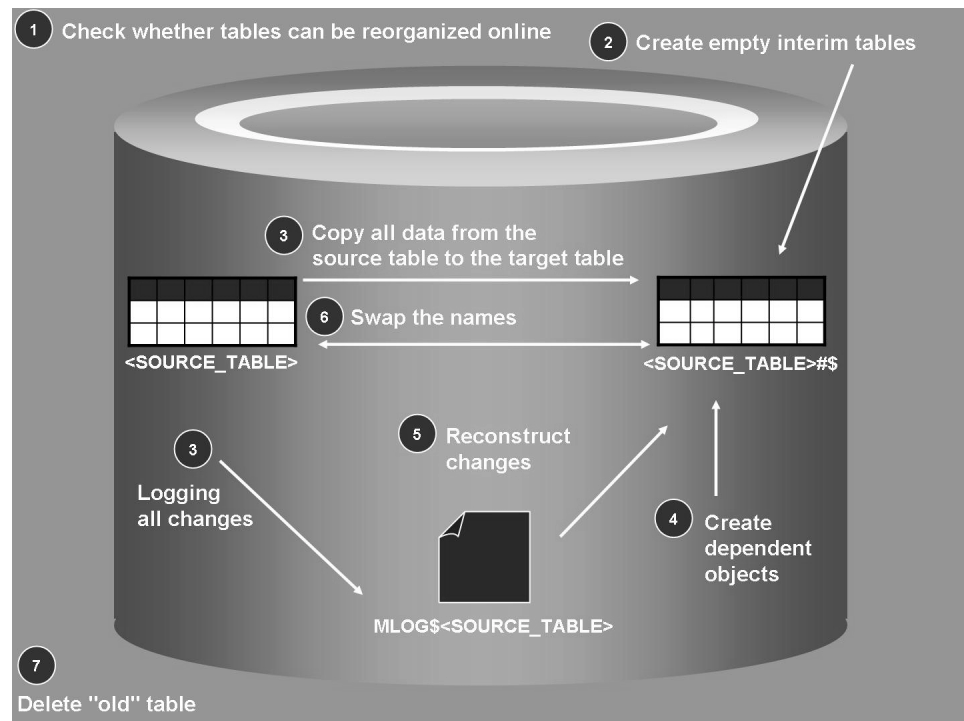


Figure 120: Online Redefinition Process

If any of the steps fails for a table, the original table is still there. BRSPACE will then drop the interim table, causing any indexes, constraints, triggers, grants, and comments on the interim table to be dropped as well.

Reorganization

Reorganize Tables

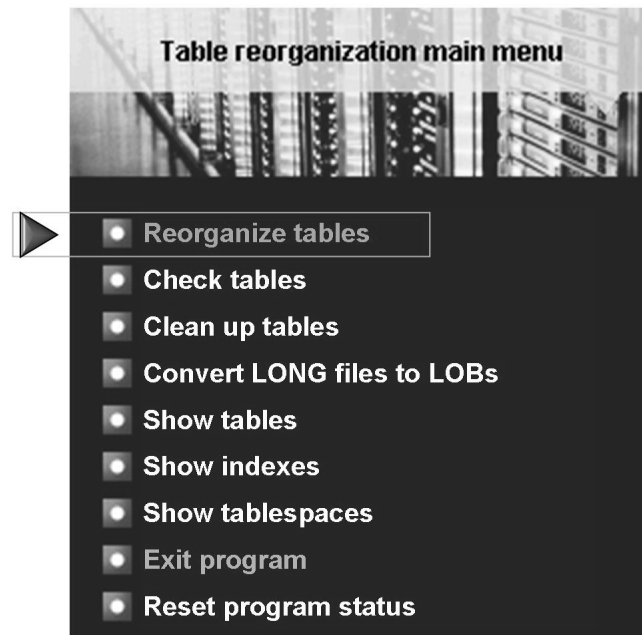


Figure 121: Reorganization of Tables Using BR*Tools

Before you start a reorganization using online table redefinition, check:

- If reorganization is used to move tables into a new tablespace, this tablespace must already exist.
- BRSPACE supports online table definition only in locally managed tablespaces.
- When moving many and/or large tables, the data files of the target tablespace have enough free space, or they should be automatically extensible and enough disk space should be provided to avoid a tablespace overflow.

To perform a reorganization using online table redefinition, start BRTOOLS or BRGUI and choose *Segment management* → *Reorganize tables*.

The menu now offered by BRTOOLS or BRGUI offers you to specify a list of tables or table names using wildcards and will go into quick mode of BRSPACE. Reorganization options are entered here. If you continue without entering any values (except the profile for BRSPACE), tables have to be selected from a list after selecting the *Reorganize tables* menu from BRSPACE.



Hint: You can also specify tablespace name instead of table names for reorganization. Then all tables from this tablespace (except tables with LONG fields) will be assumed for reorganization.



Hint: To reorganize a set of tables that cannot be matched by wildcards ([<owner.><prefix>* or *), you can use parameter `reorg_table` in `init<DBSID>.sap` to specify a list of tables to be reorganized.

After selecting the tables to be reorganized, the following menu is displayed:

```
BR0657I Input menu 353 - please check/enter input values
```

```
-----
Options for reorganization of tables: SAPT99.SKEL (degree 1)
```

```
1 ~ New destination tablespace (newts) ..... [PSAPT99NEW]
2 ~ Separate index tablespace (indts) ..... []
3 - Parallel threads (parallel) ..... [1]
4 ~ Table/index parallel degree (degree) ..... []
5 - Create DDL statements (ddl) ..... [yes]
6 ~ Category of initial extent size (initial) . []
7 ~ Sort by fields of index (sortind) ..... []
8 - Table reorganization mode (mode) ..... [online]
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
```

```
-----
BR0662I Enter your choice:
```

The parameters and their meaning are:

newts

Enter the tablespace into which the tables are to be reorganized. This tablespace must exist and be a locally managed tablespace.

If you do not specify a tablespace, the tables are reorganized within their original tablespace

indts

You only set this parameter if the tables are reorganized into another tablespace that only contains table segments. Specify the corresponding index tablespace here.

parallel

Reorganization can be parallelized using this option. A high parallelism speeds up reorganization, but will have a greater impact on performance. With low parallelism, reorganization will take longer but not affect performance as much.

As a rule of thumb, `parallel` should be set to

- One (1) when reorganization is performed while SAP is active. Reorganization will have an impact on performance and should be performed in off-peak hours.
- To a value of one to two times the number of CPUs when reorganization is performed while SAP is down.

degree

To parallelize the copy of data from the original table into the interim table. Then parallel query will be used by the *DBMS_REDEFINITION* package during the copy. You can use this option to shorten a reorganization of large tables.

ddl

To create the intermediate tables, BRSPACE creates an SQL script containing the DDL statements in `$$SAPDATA_HOME/sapreorg/<encoded_timestamp>` called `ddl.sql`. This parameter will influence the creation of the DDL script:

- If set to NO, the DDL statements will only be created internally and will not be visible, which is not recommended.
- The default setting is YES, which means that `ddl.sql` will be created and, during reorganization, automatically used for the creation of intermediate tables.
- FIRST means that `ddl.sql` will be created before reorganization and BRSPACE will stop before executing it. This enables you to view the statements and adapt them if necessary. Select *continue* to perform the reorganization.



Caution: Changing the DDL statements before execution should only be performed by an Oracle expert.

- ONLY will actually not perform a reorganization, but only create the DDL statements. Only in this case, no `#$` is appended to the object names.

initial

Category of the INITIAL extend size.

sortind

Sorting according to index fields in order to improve partially sequential accesses.

mode

By default, the Oracle DBMS_REDEFINITION package is used for the reorganization [**online**]. Instead of the Oracle DBMS_REDEFINITION package, it is also possible to use the statement ALTER TABLE [PARTITION] MOVE [**offline**]. No DDL statements are generated. This method can be quicker than an online reorganization, but locks the affected tables. The SAP system should therefore not be in operation during the process.

Rebuild Indexes

To rebuild one or more indexes online, start BRTOOLS or BRGUI and choose *Segment management* → *Rebuild indexes*. The menus are similar to those of the table reorganization. But there are three exceptions:

- There is no DDL menu item, as index reorganization is performed with the Oracle command ALTER INDEX REBUILD ONLINE and no objects have to be created manually.
- In the menu of BRTOOLS, you can enter a list of tables. If you enter tables instead of indexes here, all indexes of the specified tables will be rebuilt.
- You can enter a new tablespace name to move indexes into another tablespace during the rebuild.

Exporting and Importing Tables

BRSPACE supports reorganization of tables by export/import, but this method should only be used if online table redefinition is not possible (for example in Oracle 9i for tables containing LONG fields).

Actually, the export/import functionality of BRSPACE is just a front end to the Oracle export and import tools. In addition, to perform a reorganization with export/import, several manual steps must be performed.

To export and import data and structures, the new method Oracle Data Pump is available as of Oracle 10g (see SAP Note 1013049). Data Pump can be viewed as an extension to the EXP and IMP tool. As opposed to the classic EXP/IMP tool, the export is always performed on the server and not on the client. The main advantages for the user are the new options for parallel export/import processes and compressing metadata. This will also improve performance.

Even though Data Pump and EXP / IMP operate similarly, the technical basics are very different. This means the approaches are not compatible, and therefore no EXP dump file can be imported with IMPDP.

The Oracle Data Pump export and import utilities are supported by BRSPACE as of Version 7.00 (17) in the functions "Export tables" and "Import tables" (see SAP Note 976435).



Caution: According to Oracle recommendations, Data Pump export and import utilities should be performed as a DBA database user. To do this, use the option "-u" when you call BRSPACE, for example: `brspace -u system/<password> -f tbexport -l expdp`.

A BRSPACE reorganization based on Data Pump mainly consists of an export and import phase. Parameters, log and dump files are stored in the \$SAPDATA_HOME/sapreorg directory by default. The basic steps are:



1. Stopping the SAP System.
2. Start the BRSPACE export according to SAP Notes 976435 and 646681.
3. Create an export parameter file with BRSPACE (parfile.exp).
4. Create DIRECTORY references for the dump and log directory with BRSPACE
5. Call EXPDP with the export parameter file created above with BRSPACE.
6. Drop the created DIRECTORY references with BRSPACE.
7. Start the BRSPACE import based on the created export dump file.
8. During the import, BRSPACE now performs the same steps as for the export (see above).
9. Start the SAP System.



Caution: Only perform a reorganization with export/import when you are familiar with the procedure itself and the Oracle tools Data Pump and EXP and IMP.

Transform dictionary-managed tablespaces to locally-managed tablespaces

To transform dictionary-managed into locally managed tablespaces with Oracle 9i, perform a normal online reorganization of all tables within a tablespace into a new tablespace, with the following settings and additional steps:

1. Before the reorganization, create one new locally managed tablespace storing tables and indexes (contents: `both`). This tablespace must be large enough to store tables **and** indexes of both the old table and the old index tablespace (assuming you have a non-MCOD tablespace layout before reorganization). It is recommended to set the data files of the new tablespace autoextensible.



Hint: If you want to have separate table and index tablespaces after reorganization, create two tablespaces: one with contents `table` and one with contents `index`.

2. Start BRGUI or BRTOOLS and choose *Segment management* → *Reorganize tables*. Specify the name of the tablespaces to be reorganized, and enter the “*” character in the table field. No other options need to be changed.



Hint: If you reorganize a table tablespace that has an associated index tablespace, only enter the name of the table tablespace. All indexes of the reorganized tables will be automatically reorganized.

3. Continue and confirm the list of tables being reorganized.
4. In the Options for reorganization of tables menu, enter the name of the new tablespace. If you want to have separate table and index tablespaces after reorganization, specify the new Index tablespace created under step 1. Enter the number of parallel threads and the option for creating DDL statements (see above). Then continue.

Now the online reorganization starts.



Hint: Tables and their indexes that cannot be reorganized online (like tables with LONG fields) will stay in the old tablespaces. To move these tables into the new locally managed tablespace, you have three options:

- At the moment, you must use the export/import functionality of BRSPACE, preferably **after** online reorganization.
- As soon as offline conversion of tables containing LONG fields into LOB fields is available in BRSPACE (as of Oracle 10g), use this functionality **before** reorganization. After this procedure, these tables can be reorganized online.

See SAP Note 646681 for details, especially to find out which option is offered at the moment and for details of the export/import (if this is necessary).

Segment Shrinking

Segment shrinking offers an alternative to reorganization for defragmenting a segment and gaining more free space. The free space occupied by a table by combining the segments is retrieved.

Segment Shrinking is available as of Oracle Release 10g. To use this new function, the following prerequisites must be met:

- The table you want to shrink must occur in an ASSM tablespace.
- Tables with LONG and LONG RAW fields cannot be shrunk.
- Mapping tables and overflow segments of IOTs cannot be shrunk.
- Compressed tables cannot be shrunk.
- ROW MOVEMENT must be activated for the table to allow ROWs to be shifted to a different location within a segment. If the ROWID is used when accessing the application, activating ROW MOVEMENT can cause problems. However, this is not the default setting in an SAP environment.

Segment shrinking gains free space by combining segments. It is not possible to store a table in another tablespace during segment shrinking.

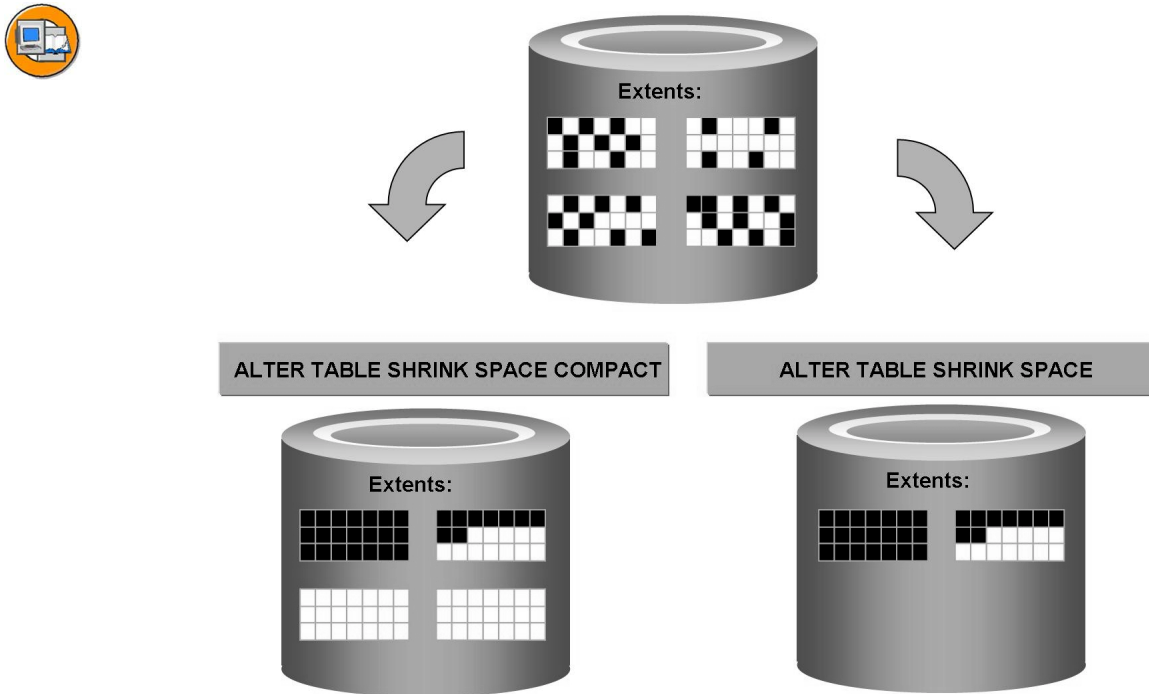


Figure 122: Segment Shrinking

Compared to offline and online reorganizations, segment shrinking offers the following advantages: Compared to an offline reorganization, there is no downtime. While an online reorganization temporarily requires twice the space, segment shrinking does not require any additional space requirements. During an online reorganization, ALL table entries are accessed, while only a section of the entries are shifted during segment shrinking. This means that the amount of redo log information during segment shrinking is less than with an online reorganization.

Segment shrinking is particularly useful if a large amount of space is lost owing to table fragmentation, causing performance problems. Segment shrinking allows you to gain space in the tablespace to be re-used for other segments, and, depending on each situation, to improve performance.

When you shrink a segment, you can also specify the options COMPACT and CASCADE:

- If you specify COMPACT, although the table is defragmented, the high water mark is not changed. This means that the excess space is not returned to the tablespace. The advantage, however, is that no locks are required.
- Without specifying CASCADE, the table is defragmented, the high water mark is adjusted, and the unused space behind the high water mark is released to be re-used by other segments in the tablespace. A temporary lock is required at the end of the runtime.
- By specifying CASCADE, not only the tables, but also dependent segments such as indexes are shrunk.



Caution: The following problems can be triggered by segment shrinking:

- Fatal corruptions can occur in tables with LOB fields. With Oracle 10.2.0.2, make sure that the Oracle bugfix from SAP Note 1021454 has been implemented before you perform segment shrinking in tables with LOB columns.
- Even if segment shrinking is mainly an online operation, table locks (TM enqueues) may occur. The best solution is to run segment shrinking only at times when no long-running transactions are active, and change the underlying table.

As of BR*Tools Release 7.00, segment shrinking can be performed using BRSPACE. For the corresponding menu, see

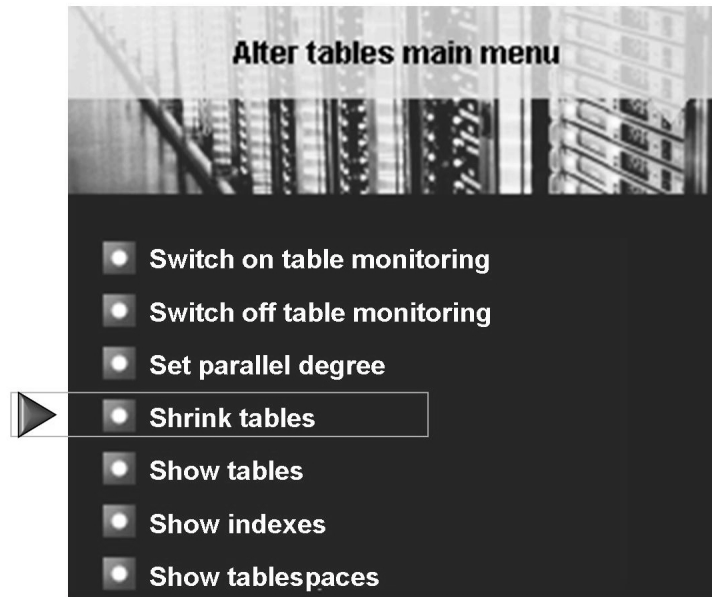


Figure 123: Segment Shrinking with BR*Tools

Exercise 12: Reorganization of Tables

Exercise Objectives

After completing this exercise, you will be able to:

- Transform dictionarymanaged into locally managed tablespaces

Business Example

Now you have learnt the advantages of locally managed tablespaces, you want to transform your dictionary managed tablespaces to locally managed tablespaces.

Task:

Convert dictionary-managed tablespaces to locally-managed tablespaces.

1. Reorganize all tables of tablespace *PSAP<DBSID>EX* into tablespace *PSAP<DBSID>NEW* (created in the last exercise).

Solution 12: Reorganization of Tables

Task:

Convert dictionary-managed tablespaces to locally-managed tablespaces.

1. Reorganize all tables of tablespace *PSAP<DBSID>EX* into tablespace *PSAP<DBSID>NEW* (created in the last exercise).
 - a) Start BRGUI or BRTOOLS and choose *Segment management* → *Reorganize tables*. In the next menu, enter *PSAP<DBSID>EX* as tablespace for reorganization, “*” to reorganize all tables in this tablespace, and *PSAP<DBSID>NEW* as the new tablespace:

```
BR0657I Input menu 91 - please check/enter input values
-----
BRSPACE options for reorganization of tables

1 - BRSPACE profile (profile) ..... [initT99.sap]
2 - Database user/password (user) .. [/]
3 ~ Reorganization action (action) . []
4 ~ Tablespace names (tablespace) .. [PSAPT99EX]
5 ~ Table owner (owner) ..... []
6 ~ Table names (table) ..... []
7 - Confirmation mode (confirm) .... [yes]
8 - Extended output (output) ..... [no]
9 - Scrolling line count (scroll) .. [20]
10 - Message language (language) .... [E]
11 - BRSPACE command line (command) . [-p initT99.sap -s 20 -l E
    -f tbreorg -s PSAPT99EX]

Standard keys: c - cont, b - back, s - stop, r - refr, h - help
-----
BR0662I Enter your choice:
c
...
...
BR0656I Choice menu 351 - please make a selection
-----
Table reorganization main menu

1 = Reorganize tables
2 - Check tables
3 - Cleanup tables
```

Continued on next page

```

4 - Convert LONG fields to LOBs
5 - Show tables
6 - Show indexes
7 - Show tablespaces
8 * Exit program
9 - Reset program status

```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

1

...

...

BR0657I Input menu 353 - please check/enter input values

Options for reorganization of tables: SAPT99.SKEL (degree 1)

```

1 ~ New destination tablespace (newts) ..... [PSAPT99NEW]
2 ~ Separate index tablespace (indts) ..... []
3 ~ Parallel threads (parallel) ..... [1]
4 ~ Table/index parallel degree (degree) ..... []
5 ~ Create DDL statements (ddl) ..... [yes]
6 ~ Category of initial extent size (initial) . []
7 ~ Sort by fields of index (sortind) ..... []
8 ~ Table reorganization mode (mode) ..... [online]

```

Standard keys: c - cont, b - back, s - stop, r - refr, h - help

BR0662I Enter your choice:

c

...

BR0280I BRSPACE time stamp: 2007-11-27 11.18.16

BR1101I Starting online table reorganization...

BR0280I BRSPACE time stamp: 2007-11-27 11.18.17

BR1124I Starting online reorganization of table SAPT99.SKEL ...

BR0280I BRSPACE time stamp: 2007-11-27 11.18.21

BR1105I Table SAPT99.SKEL reorganized successfully

...

b) Choose *Continue* until reorganization starts.



Lesson Summary

You should now be able to:

- Describe a situation in which a reorganization is necessary
- Perform a reorganization
- Transform dictionary-managed tablespaces to locally-managed tablespaces

Lesson: Housekeeping and Troubleshooting

Lesson Overview

This lesson first gives an overview of all regular activities that should be performed on the database. Some typical problems that might occur on the database, how to prevent them, and how to solve them will also be discussed.



Lesson Objectives

After completing this lesson, you will be able to:

- List the most common problems that occur in Oracle databases
- Prevent an archiver stuck and solve it when it occurs
- Solve typical problems that occur on productive Oracle databases

Business Example

You have scheduled all recommended checks and backups. Nevertheless, an archiver stuck occurs. You need to resolve the archiver stuck as soon as possible, and you want to know how an archiver stuck can be prevented in the future.

Introduction

First, the recommended checks are explained, how you can view the results, and how you should respond to warnings or errors. Performing the regular checks and viewing the results in most cases enables you to act before an error occurs, instead of waiting for the error to happen.

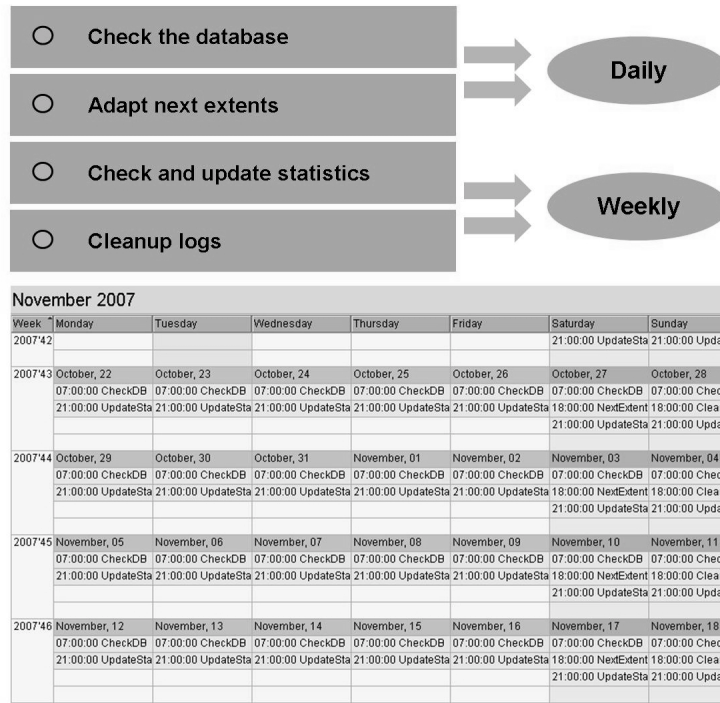


Figure 124: Regular Database Checks

Database system check

Tool

```
brconnect -u / -c -f check
```

Schedule

Planned to run daily, scheduled with the DBA Cockpit (transaction DB13), action *Check database*.

Results

Check the action log daily with the DBA Cockpit (transaction DB14).
Check all yellow or red messages created after your last check of the results.

Actions

Resolve warnings and errors.

For open messages of type **DBA** (database administration), check the error or warning. Generally, there are two solutions to solve the problem. For most warnings or errors you must check the reason for the message and then solve it. For some warnings you might decide that there is no reason to react because you do not consider it as a warning. Example: The tablespace *PSAP<SCHEMA-ID>* has an overall size of 500 GB. During the database check, the system tells you that the tablespace is 95% full, and there is 24 GB free capacity. You decide after checking the database growth with transaction DB02 that you only want to be warned if a tablespace has less than 5 GB free space. In this case you would change the check condition in table *DBCHECKORA* using transaction DB17.

For open messages of type **DBO** (database operation), check the scheduling of the corresponding action if the last operation is too old and reschedule the action. Check the log of the background job and the detailed log of the operation if it failed, solve the problem, and repeat the operation.

For open messages of type **PROF**, adapt the parameter in the Oracle profile. If you are sure that despite the warning, the current parameter is correct (for example because of an SAP EarlyWatch recommendation, or because this value is listed in SAP Note 124361), change the value of the check condition in table *DBCHECKORA* using transaction DB17.

For open messages of type **ORA**, (Oracle error messages) check the Oracle documentation (command `oerr ora <message number>` on UNIX, documentation "Oracle Error Messages" on all operating systems) which provides the possible cause and solution of the error. If you are not sure about the cause of the problem and the solution, search the note in the SAP Service Marketplace for the error message.

Check and adapt NEXT extents

Tool

```
brconnect -u / -c -f next
```

Schedule

Planned to run weekly, scheduled with the DBA Cockpit (transaction DB13), action *Adapt next events*.

Results

Check the action log weekly with the DBA Cockpit (transaction DB14).
Check all yellow or red messages created after your last check of the results.

Actions

In rare cases, BRCONNECT will adapt the NEXT extent size of certain tables that are too large, or you want to set the NEXT extent size to a value defined by yourself. In this case, adapt the profile `init<DBSID>.sap` and set the parameter `next_exclude` to exclude tables from adapting the NEXT extent size. You could also set the parameter `next_special` to set a self-defined NEXT extent size.



Hint: Do not schedule this action if all tablespaces in the database are locally managed.

Update optimizer statistics

Tool

```
brconnect -u / -c -f stats -t all
```

Schedule

Planned to run daily with Oracle 10g, scheduled with the DBA Cockpit (transaction DB13), action *Check and update optimizer statistics*.



Hint: For all other Oracle releases up to 9i, schedule a weekly run.

Results

Check the action log daily with the DBA Cockpit (transaction DB14).
Check all yellow or red messages created after your last check of the results.

Actions

Sometimes, new tables are created by developers which do not have statistics until the next planned run of `brconnect -f stats`. To create statistics of only tables that do not have statistics, run `brconnect -f stats -t missing`, or start it with transaction DB20, and choose *Global statistics → Create missing*.

Clean up old logs and traces

Tool

```
brconnect -u / -c -f cleanup
```

Schedule

Planned to run weekly, scheduled with the DBA Cockpit (transaction DB13), action *Cleanup logs*.

Results

Check the action log weekly with the DBA Cockpit (transaction DB14).
Check all yellow or red messages created after your last check of the results.

Actions

No general hints. If log files are deleted too early or too late, check the parameter `cleanup_*` in profile `init<DBSID>.sap`.

Database backup

Tool

BRBACKUP and BRARCHIVE

Schedule

Planned to run daily, scheduled with the DBA Cockpit (transaction DB13). In most cases, planning the action *Whole database online + redo log backup* is sufficient, as here BRBACKUP and BRARCHIVE are executed in one run. When choosing any other backup strategy, make sure:

- For any incremental or partial backup, you have four full backups within your backup cycle.
- For any whole backup without redo logs, incremental, or partial backup, you have scheduled a daily redo log backup.

Results

Check the action log daily with the DBA Cockpit (transaction DB14). Check all yellow or red messages created after your last check of the results.

Actions

No general hints. Check the log files of BRBACKUP and BRARCHIVE.

Database Support without ABAP Schema

As of SAP Web Application Server Java 6.40 SR 1, the tables SDBAH and SDBAD are created in the Java database schema (SAP<Schema-ID>DB) as standard. This completes the requirements for using BR*Tools for pure Java databases (without ABAP schema), which means you can save and restore Java databases with the BR*Tools.

The lack of the DBSTATC and DBCHECKORA tables in this environment, however, means that there are practically no Customizing options for updating the database statistics or for the database check. This restriction has now been resolved in BR*Tools 7.00. You can also use these tools with Oracle 9.2. Oracle Instant Client 10g must be installed on the database server (for more information, see SAP Note 849483).

In BR*Tools 7.00, two parameters were introduced in `init<DBSID>.sap`, which assume the function of the tables `DBSTATC` and `DBCHECKORA` (see SAP Note 892294):

- `stats_special`, for tables with specific statistics handling.
- `check_cond`, for Customizing the check conditions



Hint: These new parameters and their options are described in detail in SAP Note 892294. This note also contains a copy template for the parameter `check_cond` according to the check conditions from the `DBSTATC` table.

Typical Problems

Even when monitoring the database regularly and performing all necessary administrative activities, sometimes problems occur.

Archiver Stuck

An archiver stuck is a situation in which the archiver instance process (ARC0) cannot copy an online redo log file to the archive directory (usually `$SAPDATA_HOME/oraarch`) **and** the online redo log file to be copied needs to be overwritten by the log writer (LGWR).

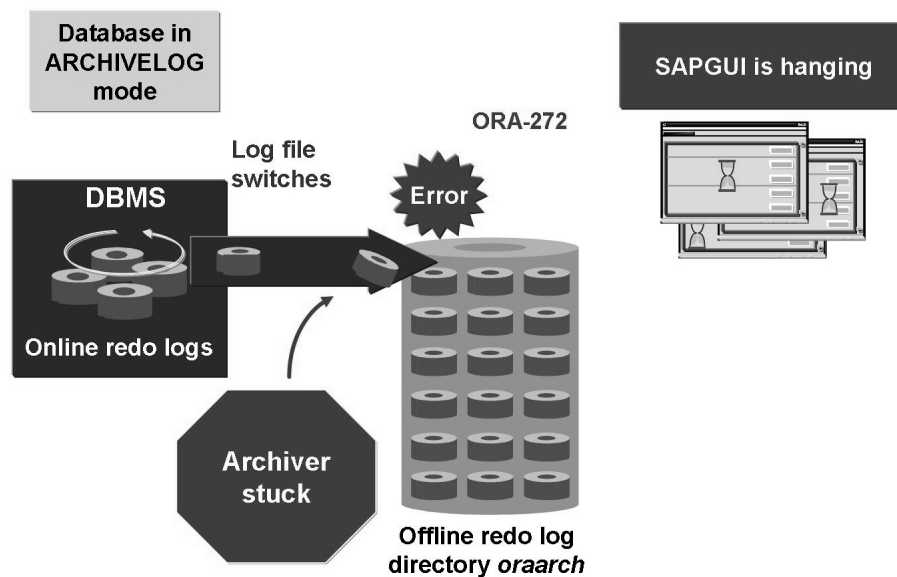


Figure 125: Archiver Stuck

When an archiver stuck occurs, the database instance is not shut down, but does not allow any changes. In the SAP system, users will see an hourglass as long as the archiver is stuck, and Oracle will write an error message into its alert log file and create a trace file in `$ORACLE_HOME/saptrace/background`.

The reason for this error is usually that the disk holding the archiving directory is full because:

- There was high database activity, for example due to SAP data archiving, so an exceptional high number of archived redo logs was written.
- One or more runs of BRARCHIVE failed, for example because there is a problem with a tape station that cannot be repaired in a short time frame. Because BRARCHIVE would normally delete archived redo logs that have been saved successfully, the archive directory fills up with more than usual.

You can never be 100% sure that you will never get an archiver stuck, because you will never have unlimited disk space. Therefore you should be prepared for it and know how to solve it. Nevertheless, there are some prerequisites that will help an archiver stuck occur more rarely, or help solve an archiver stuck more easily:



- The archive directory should have as much disk space as possible. The minimum is three times the space you need for the archived redo logs per day, because the default action of BRARCHIVE is to backup the archived redo logs twice before deleting them, and you need some extra space to store the archived redo logs when you have higher database activity. To be able to perform a recovery over a long time frame, a minimum of 100 offline redolog files should fit into the archive directory.



Hint: The larger the disk space in the archive directory, the smaller the chance of getting an archiver stuck!

- Create a dummy file of about 10 times the size of an online redo log file in the archive directory with the name `_delete_me_on_archiver_stuck`. An archiver stuck cannot be avoided completely; when it occurs, you simply delete the dummy file, giving you some time to run BRARCHIVE to save and delete old archived redo logs.



Caution: Never delete archived redo logs without having them backed up, as a full recovery from any backup created up to now would not be possible!

- The archive log (defined by the Oracle parameter `log_archive_dest`) should not be located on the same hard disk as the directory in which BRARCHIVE writes its log files (`$SAPARCH`, or, if not set, `$SAPDATA_HOME/saparch`). If the archive log is full, BRARCHIVE may fail to save and delete existing archived redo log files because BRARCHIVE cannot write to its own log file.



Hint: Up to SAP R/3 4.6C, the SAP installation tools chose `$SAPDATA_HOME/saparch` to store archived log files **and** the BRARCHIVE log files. You should manually change it to `oraarch` in this situation.

To store archived log files on a disk different from the BRARCHIVE log files, perform the following steps:

1. Check the current `log_archive_dest` parameter and make a note of the complete path. To do so, start BRTOOLS or BRGUI and choose *Instance Management* → *Show database parameters*.
2. Run `brarchive -cds` twice to save and delete all existing archived redo log files.
3. Create a new directory, `orarch`, on a different disk.
 - In Windows, on the other disk, the complete directory tree `<new disk>:\oracle\<DBSID>\oraarch` must be created.
 - In UNIX, create the directory `/oracle/<DBSID>/oraarch`, unmount `/oracle/<DBSID>/saparch`, change the mount definition in the file systems table (usually `/etc/fstab`, the file system which was mounted to `/oracle/<DBSID>/saparch` should now be mounted to `/oracle/<DBSID>/oraarch`) and mount `orarch` with the command `mount /oracle/<DBSID>/oraarch`. Now copy the old log files back to `saparch`: `cp /oracle/<DBSID>/oraarch/* /oracle/<SID>/saparch`.
4. Change the `log_archive_dest` parameter. Only change the path; the last part of the original `log_archive_dest` (see above) is not a directory, but the prefix for the names of the archived redo logs. Use the command `brspace -c force -f dbparam -p log_archive_dest -v <new drive>:\oracle\<SID>\oraarch\<SID>arch` on Windows and `brspace -c force -f dbparam -p log_archive_dest -v /oracle/<SID>oraarch/<SID>arch` on UNIX.
5. To check if everything is okay, perform a manual log switch with `brspace -c force -f dbalter -a switchlog` and check if a new archived redo log file was created in the new `orarch` directory.



- Delete the dummy file

When an archiver stuck occurs, first delete the dummy file. After deleting the dummy file, the archiver stuck is immediately resolved and the database continues to work. After that, perform a `brarchive -cds` twice. The second run of BRARCHIVE is necessary, because the first run will only delete archived redo logs which were already backed up the day before and, therefore, will not release much disk space.

- Changing archive directory

BRARCHIVE 7.00 is able to find and save offline redo log files after changing the archive directory, which offers the following simple procedure for resolving an Archiver Stuck problem:

- Use BRSPACE to change the value of the Oracle parameter `log_archive_dest[_1|_2]` so that it points to a new archive directory with enough free space.
- Start BRARCHIVE to save and delete offline redo log files. Make sure that the offline redo logs can be saved twice on different media. To do this, run `brarchive -cds` twice.
- Then change the value of the parameter `log_archive_dest` back to the original value.

Online Backup Crashed

During an online backup, tablespaces are set into backup mode to enable proper recovery whenever a data file from an online backup needs to be restored. BRBACKUP will make sure that after an online backup, the backup mode of all tablespaces backed up is turned off again.

If BRBACKUP crashes, one or more tablespaces remain in backup mode. Furthermore, a lock file of BRBACKUP can prevent further backups to start. In this situation:

- The database cannot be shut down to a consistent state; only a shutdown abort is possible.



Caution: In this case, do not shut down the database with Shutdown Abort. If you do so, the next startup will not work and additional actions will need to be performed.

- Any other BRBACKUP can fail.
- A database crash or shutdown abort would leave the database in a state where a normal startup fails.

While a tablespace is in backup mode, `brconnect -f check` issues the following warning: `WARNING, type: TABLESPACE_IN_BACKUP, object: PSAPT00EX`. To check if a tablespace is currently in backup mode, you can use the faster (and cheaper) command `brspace -c force -f dbshow -t tslist`. Check the *Back* column to see if any tablespace is in backup mode.

If any tablespace is in backup mode, but you expect no BRBACKUP to be running, perform the following steps to solve the problem:

1. Check if BRBACKUP is running using the `ps` command on UNIX or the Task Manager on Windows. Only continue, if no BRBACKUP is running.
2. Start BRTOOLS or BRGUI and choose *Space management* → *Alter tablespace* → *Reset backup status*. Choose *continue* twice to get a list of all tablespaces in backup mode.
3. Enter 0 to select all tablespaces and choose *continue* to turn backup mode off.
4. Delete the lock file `$SAPDATA_HOME/sapbackup/.lock.brb`.
5. Run a test backup with BRBACKUP to check if everything is okay.

Sometimes it happens that the whole database or server crashes during an online backup, for example when a reboot is executed while BRBACKUP is running. In this case, a normal startup is not possible and you will see the following error message when starting the database:

```
...
SQL> ORACLE instance started.

Total System Global Area  48307140 bytes
Fixed Size                 453572 bytes
Variable Size             41943040 bytes
Database Buffers          5734400 bytes
Redo Buffers              176128 bytes
Database mounted.
ORA-01113: file 5 needs media recovery
ORA-01110: data file 5: 'D:\ORACLE\T99\SAPDATA3\T99_1\T99.DATA1'
...
```

You can solve this problem by performing a recovery on the database using BRRECOVER. If no other problem has occurred, an unattended recovery using the command `brrecover -c force -t complete` will solve the problem without further prompts. To perform the recovery step by step, start `brrecover` without further options and proceed as described in lesson Restore and Recovery.



Hint: The problem can also be solved without performing a recovery, which can take a long time depending on the database activity and the duration of the tablespace being in backup mode. To avoid a long recovery, use SQL*Plus and startup the database into mount state. Then execute the command `ALTER DATABASE END BACKUP` and open the database.



Caution: The command `ALTER DATABASE END BACKUP` must only be used in this case, when no data file has been restored! Never use this command after having restored any data file. Doing so could bring your database into an inconsistent and unusable state!

Exercise 13: Housekeeping and Troubleshooting

Exercise Objectives

After completing this exercise, you will be able to:

- Solve the situation when the database server crashes during an online backup

Business Example

During an online backup, the server crashed and you have to get the database back online.

Task:

Simulate a server crash during online backup and get the database running again.

1. Simulate a server crash during online backup by setting tablespace *PSAP<DBSID>* in backup mode and shut down the database with the *abort* option. Try to start the database and check the error message.
2. Now get the database running again.

Solution 13: Housekeeping and Troubleshooting

Task:

Simulate a server crash during online backup and get the database running again.

1. Simulate a server crash during online backup by setting tablespace *PSAP<DBSID>* in backup mode and shut down the database with the *abort* option. Try to start the database and check the error message.

- a) Start BRGUI or BRTOOLS and choose *Space management* → *Alter tablespace* → *Set backup status*. Then select *PSAP<DBSID>* and continue to put it in backup mode.

```
BR0657I Input menu 313 - please check/enter input values
```

```
-----
Options for alter of tablespace PSAPT99
```

```

1 * Current tablespace status (status) . [ONLINE]
2 * Current backup status (backup) ..... [NO]
3 * Alter tablespace action (action) ... [begback]
4 # Set offline mode (mode) ..... [normal]
5 # New tablespace name (name) ..... []
6 # Force tablespace alter (force) ..... [no]
7 - SQL command (command) ..... [alter tablespace PSAPT99
    begin backup]
```

```
Standard keys: c - cont, b - back, s - stop, r - refr, h - help
```

```
-----
BR0662I Enter your choice:
```

- b) Shut down the database with the *abort* option. To do so, start BRTOOLS or BRGUI and choose *Instance Management* → *Shut down database*. Then select *abort* as the shutdown mode.
- c) Try to start the database again by starting BRGUI or BRTOOLS and choosing *Instance management* → *Start up database*. The following error message will be displayed:

```
BR0304I Starting and opening database instance T99 ...
```

```
BR0278E Command output of 'g:\oracle\DEV\102\BIN\sqlplus':
```

```
SQL*Plus: Release 10.2.0.2.0 - Production on Tue Nov 27 11:43:12 2007
```

Continued on next page

Copyright (c) 1982, 2005, Oracle. All Rights Reserved.

SQL> Connected to an idle instance.

SQL>

SQL> ORACLE instance started.

Total System Global Area 134217728 bytes

Fixed Size 2150000 bytes

Variable Size 113127824 bytes

Database Buffers 16777216 bytes

Redo Buffers 2162688 bytes

Database mounted.

ORA-01113: file 4 needs media recovery

ORA-01110: data file 4: 'G:\ORACLE\T99\SAPDATA3\T99_1\T99.DATA1'

SQL> Disconnected from Oracle Database 10g Enterprise Edition Release
10.2.0.2.0 - 64bit Production

With the Partitioning, OLAP and Data Mining options

BR0280I BRSPACE time stamp: 2007-11-27 11.43.19

BR0279E Return code from 'g:\oracle\DEV\102\BIN\sqlplus': 0

BR0302E SQLPLUS call for database instance T99 failed

BR0306E Start and open of database instance T99 failed

2. Now get the database running again.

- a) The recommended way to get the database running again is to use BRRECOVER, using the scenario *Complete database recovery*. To recover the database, start BRGUI or BRTOOLS and choose *Restore and recovery* → *Complete database recovery*.
- b) Watch the actions performed by BRRECOVER. The recovery should complete successfully without further input by pressing *Continue* at any prompt.



Lesson Summary

You should now be able to:

- List the most common problems that occur in Oracle databases
- Prevent an archiver stuck and solve it when it occurs
- Solve typical problems that occur on productive Oracle databases



Unit Summary

You should now be able to:

- Create new tablespaces
- Extend existing tablespaces
- Move or rename data files
- Additional database space management options
- Describe a situation in which a reorganization is necessary
- Perform a reorganization
- Transform dictionary-managed tablespaces to locally-managed tablespaces
- List the most common problems that occur in Oracle databases
- Prevent an archiver stuck and solve it when it occurs
- Solve typical problems that occur on productive Oracle databases

Unit 5

Introduction to Oracle cache management

Unit Overview

This unit describes the Oracle architecture, in particular the memory management. Focus is put on the Dynamic System Global Area and Automatic Program Global Area.



Unit Objectives

After completing this unit, you will be able to:

- Explain the concepts of Oracle cache management
- Describe the components and the architecture of the System Global Area
- Describe Oracle's dynamic storage management, known as the dynamic System Global Area
- Discuss how Oracle processes an SQL statement
- Describe the advantages and the features of the automatic Program Global Area

Unit Contents

Lesson: Oracle System Global Area	364
Exercise 14: Activation of Dynamic System Global Area	381
Lesson: Oracle Program Global Area	386
Exercise 15: Activation of Automatic Program Global Area	397

Lesson: Oracle System Global Area

Lesson Overview

This lesson briefly explains Oracle architecture. Focus is put on Oracle cache management, especially on the Oracle dynamic System Global Area and the shared SQL area.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain the concepts of Oracle cache management
- Describe the components and the architecture of the System Global Area
- Describe Oracle's dynamic storage management, known as the dynamic System Global Area
- Discuss how Oracle processes an SQL statement

Business Example

You want to analyze the performance of your Oracle database. As a prerequisite you need to know how Oracle cache management works and how Oracle processes SQL statements.

Introduction to Oracle Architecture

An Oracle server is a relational database management system (RDBMS), which is a server that manages relational data in a database.

An RDBMS (or a database server) is able to:

- Manage large amounts of data in a multi-user environment so that many users can concurrently access the same data
- Maintain relationships between data
- Control access to data in terms of security, using its own user authorization concept
- Recover data to a point of known consistency in the event of a system failure
- Deliver high performance to process data requests

In an SAP system, the only interactive user connecting to the database server should be the database administrator. Application data processing is almost exclusively initiated by work processes of SAP instances in the role of database clients.

Terminology

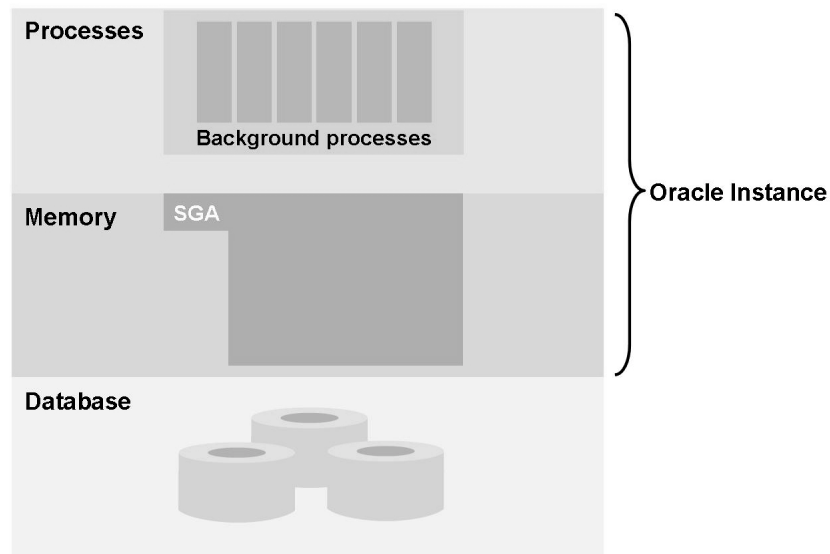


Figure 126: Database Terminology

database

An Oracle database is a collection of data, logically treated as a unit. Physically, the data is stored in one or more data files on disks.

Oracle manages database data in logical units called **tablespaces**. A database object, such as a table, is always created in a particular tablespace. A tablespace consists of one or more data files.

instance

Because the database is only a passive part of a database server, some processes and memory structures are needed to access the data and manage the database. The combination of Oracle (background) processes and memory buffers is called an Oracle **instance**.

Every running Oracle database is associated with an Oracle instance. Moreover, every Oracle database needs its own instance.



Hint: Using real application clusters (RAC), one database is served by two or more instances.

SGA

Every time an Oracle instance is started, a shared memory region called the **System Global Area (SGA)** is allocated. The SGA allocated by an Oracle instance can only be accessed by the processes of this instance. This means that each instance has its own SGA. The SGA contains copies of data and control information for the corresponding Oracle instance. When an instance shuts down, it deallocates the SGA.

processes

Every time an Oracle instance is started, Oracle background **processes** are started. When an instance shuts down, the processes are stopped.



Hint: In a UNIX environment, you can identify Oracle processes as individual system processes, while on Windows platform they are implemented as threads run within one common Oracle operating system process, *oracle.exe*. These processes are not visible when you list processes at operating system level.

system identifier (DBSID)

Every database is uniquely identified in the network by its **system identifier**. On SAP systems, the system identifier must consist of exactly three characters. The first character must be an uppercase letter, while the other two can be uppercase letters or digits. Because the term “system identifier” is also used for SAP systems, we distinguish between the database system identifier, or DBSID, and the SAP system identifier, or SAPSID.

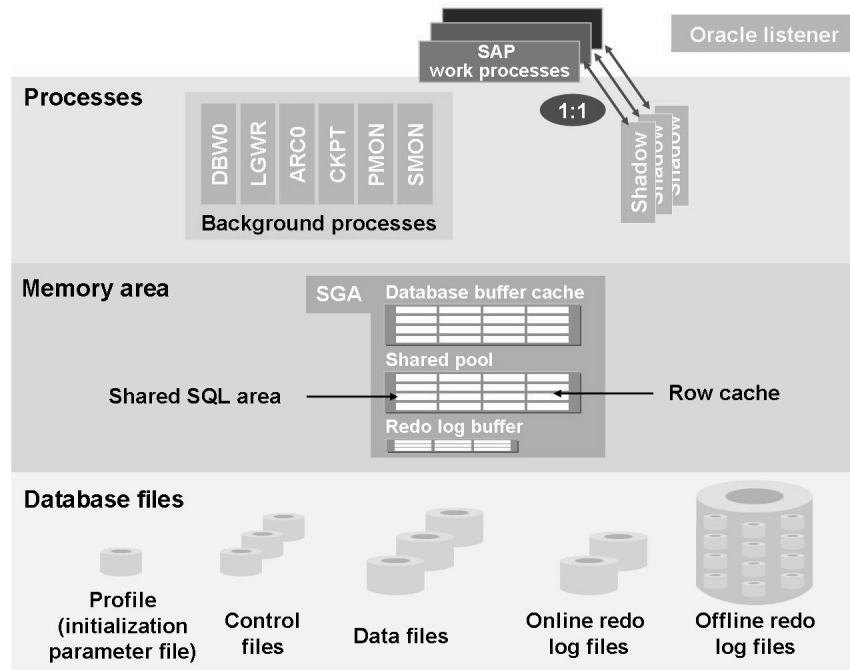


Figure 127: Architecture Overview

When an Oracle instance starts, a special process called the (network) **listener** process opens and establishes a communication pathway through which the database clients connect to the instance.

➔ **Note:** The listener process is not part of an Oracle instance; it is part of networking processes that work with Oracle.

In SAP installations, the dedicated server configuration is used. When a work process makes a connection request, the listener creates a dedicated server process and establishes an appropriate connection.

- The separate server process created on behalf of each work process (generally, of a user process) is called a **shadow process**.
- To handle database requests for various SAP system users, a work process communicates with its corresponding shadow process.
- When a work process has lost its connection to the database system, it automatically reconnects after the database server is available again and a database request is to be processed.

Oracle **background processes** perform various tasks required for the database management system to function properly.

Database data is permanently stored in data files on disks to accelerate read and write access to data. It is cached in the **database buffer** cache in SGA.

The Oracle database management system holds the executable SQL statements in the **shared SQL area** (also called the **shared cursor cache**), which is part of the shared pool allocated in SGA. Another part of the shared pool called the **row cache** caches Oracle data dictionary information.

Oracle Memory Areas

There is a distinction between shared memory areas that can be called by all Oracle processes and process-local memory, which is assigned to exactly one process in each case.

- **System Global Area (SGA):** shared memory
 - **Buffer pool** (data buffer or buffer cache): Buffer for the data blocks that are read from disk and stored in memory in the buffer pool. All user processes connected to the database share access to the buffer pool.
 - **Shared pool:** Buffer for parsed SQL statements that can be reused (shared SQL area or shared cursor cache) and information from the oracle data dictionary, such as user account data, table and index descriptions, and privileges (dictionary cache or row cache).
 - **Java pool:** The Java pool handles all session-specific Java code and data within the Java Virtual Machine. The Java Pool is not used in the SAP standard.
 - **Large pool:** This optional area is used for special data such as large I/O requests for various server processes (for example, when using a multi-threaded server, RMAN with several I/O slaves, or activating PARALLEL_AUTOMATIC_TUNING).
 - **Streams pool:** Pool for Oracle streams (Oracle 10g or later). Streams are, by default, not used in the SAP environment.
 - **Redo buffer:** This buffer caches redo information, which is used for instance recovery. This information is written to the physical redo log files.
- **Program Global Area (PGA):** process-local memory

This buffer is the part of the memory that is privately assigned to an Oracle server process (shadow process) and that contains data and information (for example, open cursors) that only this server process needs to process queries and commands. Examples of these kinds of memory-intensive operations are sortings, hash joins, bitmap operations, and other temporary local memory requirements (for example, during the parsing of SQL statements).

If the buffer is no longer sufficient, the PSAPTEMP temporary tablespace is also used.

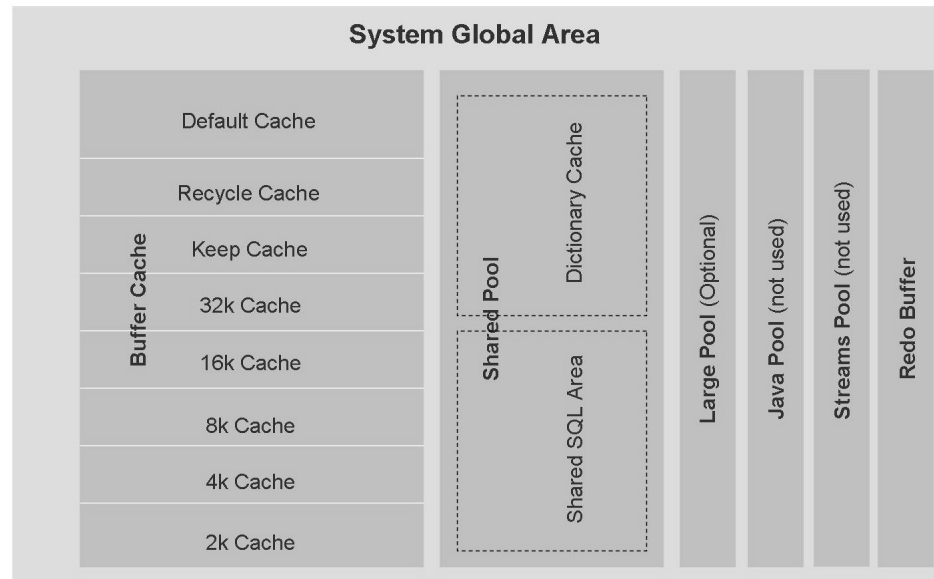


Figure 128: System Global Area: Overview

The sizes of these memory areas are defined by the following parameters:

- **SGA:** The size is determined indirectly from the size of the contained memory areas.
 - Buffer pool: DB_BLOCK_BUFFERS (unit: blocks) or DB_CACHE_SIZE when you use the dynamic SGA
 - Shared pool: SHARED_POOL_SIZE
 - Java pool: JAVA_POOL_SIZE
 - Large pool: LARGE_POOL_SIZE
 - Streams pool (Oracle 10g or later): STREAMS_POOL_SIZE
 - Redo buffer: LOG_BUFFER

In addition, in the context of the dynamic SGA, you can define parameter SGA_MAX_SIZE, which sets an upper limit for the total size of the SGA. In general, you can only increase the size of parameters, such as DB_CACHE_SIZE or SHARED_POOL_SIZE, up to the size defined by SGA_MAX_SIZE.

- **PGA:** The PGA allocation is dynamic and can be affected by the parameters SORT_AREA_SIZE, HASH_AREA_SIZE, BITMAP_MERGE_AREA_SIZE, and CREATE_BITMAP_AREA_SIZE or PGA_AGGREGATE_TARGET when you use automatic PGA administration.

Unless otherwise specified, parameters are specified in bytes.

In an SAP system useful sizes for the individual memory areas are as follows.

- **Buffer pool:** The buffer pool size generally has the greatest influence on database performance. A larger buffer pool means that the system needs to carry out fewer time-consuming disk accesses. Ensure that the data buffer quality is greater than 94%.



Note: The larger the buffer pool, the better (provided that sufficient physical memory is available). The following side effects may occur with a large buffer pool:

Oracle 9i or lower: The BEGIN BACKUP times are proportional to the size of the buffer pool.

Oracle 10.2.0.3 or higher: The runtimes of operations that involve a large amount of data (for example, imports) may be much longer with large buffer pools due to block list scans in Oracle.

- **Shared pool:** The shared pool should be at least 400 MB in size. Also, you should fulfill the approximations contained in SAP Note 618868. Also refer to the recommendations contained in SAP Note 690241. You should not configure the shared pool to a size much greater than that recommended in SAP Note 690241, as this can cause serious performance problems.
- **Java pool:** If no Java components are used, the Java pool can have a size of 0. In the SAP standard system, Java is not used.
- **Large pool:** You do not have to configure the large pool explicitly. Memory consumption is small in comparison with the other memory areas.
- **Streams pool:** The streams pool can be defined with size 0 since streams are, by default, not used in the SAP environment. Oracle automatically allocates 10% of the size of the shared pool in the buffer pool for a streams pool configured with size 0. This implicit reduction of the buffer pool must be taken into account if you want to use Oracle Data Pump because streams are used as part of Data Pump (see SAP Note 1013049).
- **Redo buffer:** A good value for the redo buffer is 1 MB, which should never be exceeded.
- **Program Global Area (PGA):** The larger you set the *_AREA_SIZE parameter or PGA_AGGREGATE_TARGET, the more PGA operations can be performed in memory.

Dynamic System Global Area

Up to and including Oracle 8i (8.1.7.x), both the SGA as a whole and the memory areas existing within the SGA were fixed with regard to their size when the instance was started. The database had to be restarted for SGA configuration changes. With

Oracle 9i or higher, most of the buffer areas in the SGA can be changed dynamically (enlarged and reduced) and thereby adjusted, for example, to be optimal for different workloads. This feature is the start of an ongoing development of the Oracle server to be able to make changes to the SGA configuration or to allow them to be carried out automatically - without stopping the instance.

This feature is also complemented by a **buffer cache advisor**. This mechanism allows the performance behavior of the instance to be predicted with different buffer cache sizes.



Hint: Multiple block sizes, or tablespaces with different block sizes (2k, 4k, 8k, 16k, 32k), are also possible with Oracle > 9i. The dynamic SGA feature is a prerequisite for multiple block sizes. We cannot yet say if multiple block sizes will be supported in the SAP environment. It is therefore recommended that you do not use this in an SAP system.

The use of variously configured SGAs is common and has, until now, only been achieved by restarting the instance, therefore involving a short system downtime. The dynamic SGA feature now allows a reconfiguration of the SGA during runtime (for example when the load profile is changed or if there is less physical memory available).

The dynamic SGA Feature offers the following advantages:



- Better options to adjust the SGA dynamically to system requirements.
- The dynamic change of buffer cache and shared pool (without restart).
- Only the SGA total size (SGA_MAX_SIZE) is fixed; within this size the other buffer areas can be changed dynamically (they can be enlarged and reduced).
- Oracle-internal cache advisor to adjust the size of the SGA optimally.
- The compiling of data for the cache advisor can be switched on and off.
- Different loads could be examined and, therefore, different SGA profiles could be created.
- Suitable for single-instance and RAC

Using the Dynamic SGA

To use this new feature, you must use new parameters.



- **SGA_MAX_SIZE**

This static parameter defines the maximum size (in bytes) up to which the SGA can increase dynamically. The sizes for the buffer cache, shared pool, and large pool can be adjusted at runtime, as long as the total of its sizes including the other components (fixed SGA, variable, SGA, and redo buffers) do not exceed the limit of SGA_MAX_SIZE. This parameter serves primarily to prevent oversizing of the SGA and paging.

If required, the SGA increases in units (granules). These granules are continuous areas of memory. With most platforms, the SGA increases up to 128 MB in 4 MB steps, then in steps of 16 MB (8 MB in Windows 32 bit). Oracle rounds off a specified size of an SGA component, possibly also to the next complete granule size. If SGA_MAX_SIZE is not set explicitly, Oracle sets SGA_MAX_SIZE as a default value to the total of all SGA components when the instance is started. It follows that the SGA cannot be larger than during the start; rather, it can only be smaller. It would therefore be useful to set this parameter explicitly onto a useful maximum value, up to which the SGA can grow dynamically - without a system downtime, without paging occurring.

- **DB_CACHE_SIZE**

Setting this parameter activates the dynamic SGA. It defines the size of the buffer cache and thereby replaces the previous parameter, DB_BLOCK_BUFFERS, which has become obsolete. DB_BLOCK_BUFFERS cannot be used simultaneously with SGA_MAX_SIZE or DB_CACHE_SIZE. Otherwise, you receive the error message, ORA-00381: cannot use both new and old parameters for buffer cache size specification..

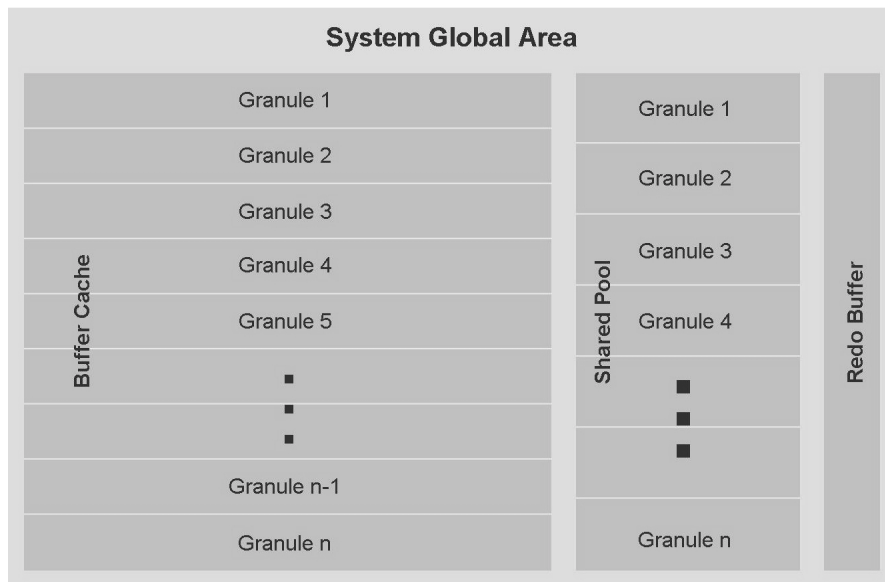


Figure 129: SGA Memory Allocation

The following parameters cannot be combined with dynamic SGA:

- DB_BLOCK_BUFFERS
- BUFFER_POOL_KEEP
- BUFFER_POOL_RECYCLE

The following areas of the SGA can be changed dynamically with Oracle > 9.2 if the SGA is configured dynamically:

- Buffer cache: DB_CACHE_SIZE
- Shared pool : SHARED_POOL_SIZE
- Large pool : LARGE_POOL_SIZE

The following cannot be changed dynamically or are only changeable by restarting the instance:

- Maximum SGA size: SGA_MAX_SIZE
- Redo log buffer: LOG_BUFFER
- Java pool: JAVA_POOL_SIZE



Hint: Dynamic SGA cannot be combined with the extended storage management (Address Windowing Extensions (AWE)) of Microsoft Windows 2000 (32 bit). For Oracle systems on Windows, which require correspondingly large SGAs, the 64-bit architecture (Windows on IA64) should be taken into consideration.

Automatic Shared Memory Management

As of Oracle 10g, parameter SGA_TARGET is also available. You can set this instead of parameters DB_CACHE_SIZE, SHARED_POOL_SIZE, JAVA_POOL_SIZE, LARGE_POOL_SIZE, and STREAMS_POOL_SIZE. If necessary, Oracle configures and then adjusts the individual SGA components within SGA_TARGET. In this context, when you stop the database, double-underscore parameters are also written to the Oracle profile to make a note of the current size of the individual memory areas (for example, __SHARED_POOL_SIZE, __DB_CACHE_SIZE). However, this dynamic adjustment of the areas can be problematic if one of the areas increases significantly in size and, as a result, other areas decrease significantly in size. Therefore, SAP recommends that you use SGA_TARGET (if you use it at all) if DB_CACHE_SIZE and SHARED_POOL_SIZE are set to sufficiently large values. In connection with SGA_TARGET, these parameters specify lower limits for the area values, below which the values cannot fall.



Caution: It is not yet possible to use ASMM (Oracle parameter SGA_TARGET) with SAP. This new feature is planned for the future.

Shared Pool

The shared pool (shared memory area in the SGA) is used by Oracle to hold several key memory structures. Most important among these are the data dictionary cache and the shared SQL area.

The data dictionary cache contains information on Oracle objects, for example:

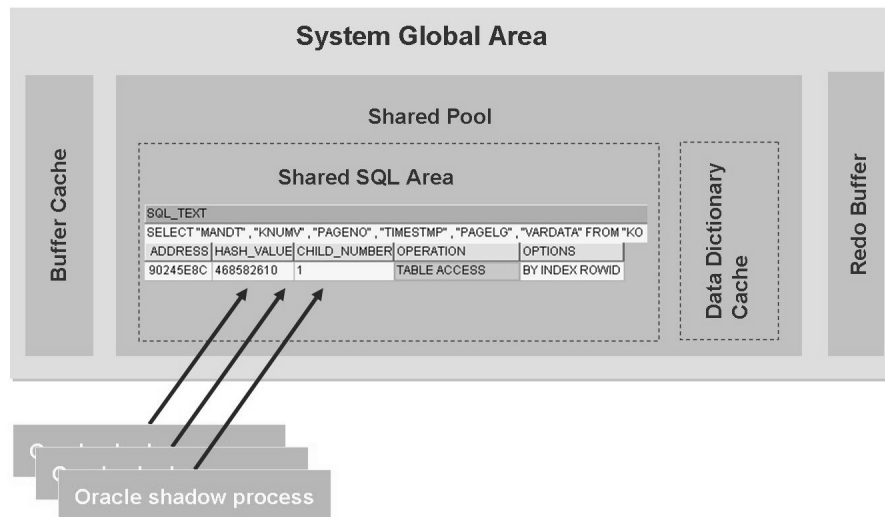
- Naming
- Definition
- Access

The shared pool is regularly referenced by Oracle itself, as well as some application programs and database users.

The shared SQL area, also known as a shared cursor cache or library cache, is a memory area that contains the parsed representation of SQL statements. Since a certain amount of system overhead is needed to parse an SQL statement, the ability to reuse statements already in memory can add a significant performance advantage.

Shared SQL Area

When application code is run, Oracle attempts to reuse existing code if it has been previously executed and can be shared. The shared SQL area is a shared memory structure in the database system that contains the parsed SQL statements and execution plans. Each SQL statement is only parsed once and then shared across all Oracle processes. If the parsed representation of the statement exists in the shared SQL area and it can be shared, then Oracle reuses the existing code. This is known as a soft parse, or a library cache hit.



- SQL statements are shared across all Oracle shadow processes.
- Cost statistics are collected for each statement.

Figure 130: Shared SQL Area

If an application makes a parse call for an SQL statement, and the parsed representation of the statement does not already exist in the library cache, then Oracle parses the statement and stores the parsed form in the shared pool. This is a hard parse. You might be able to reduce library cache misses on parse calls by ensuring that all shareable SQL statements are in the shared pool whenever possible.

If an application makes an execute call for a SQL statement, and the executable portion of the previously built SQL statement has been aged out (that is, deallocated) from the shared SQL area to make room for another statement, then Oracle implicitly re-parses the statement, creating a new shared SQL area, and executes it. This also results in a hard parse. Usually, you can reduce library cache misses on execution calls by allocating more memory to the shared SQL area.

To perform a hard parse, Oracle uses more resources than during a soft parse. Resources used for a soft parse include CPU and library cache latch gets. Resources required for a hard parse include additional CPU, library cache latch gets, and shared pool latch gets. Latches are Oracle-internal locks that protect memory structures in the SGA.

Oracle reuses parsed statements and collects cost statistics for each statement. Because all Oracle processes share the SQL statements, you can monitor the global cost of statements using the shared SQL area.

Oracle processes SQL statements in two phases:

- During the parse phase, an SQL statement is compared to other statements in the shared SQL area. If a match is found, the parsed statement is reused. If a match is not found, the SQL statement is stored in the shared SQL area with the access path.
- During the data retrieval phase, Oracle locates the required tables and loads the appropriate data blocks into the data buffer. Oracle then performs the necessary operations on the data. The data retrieval phase consists of two steps: open and fetch.

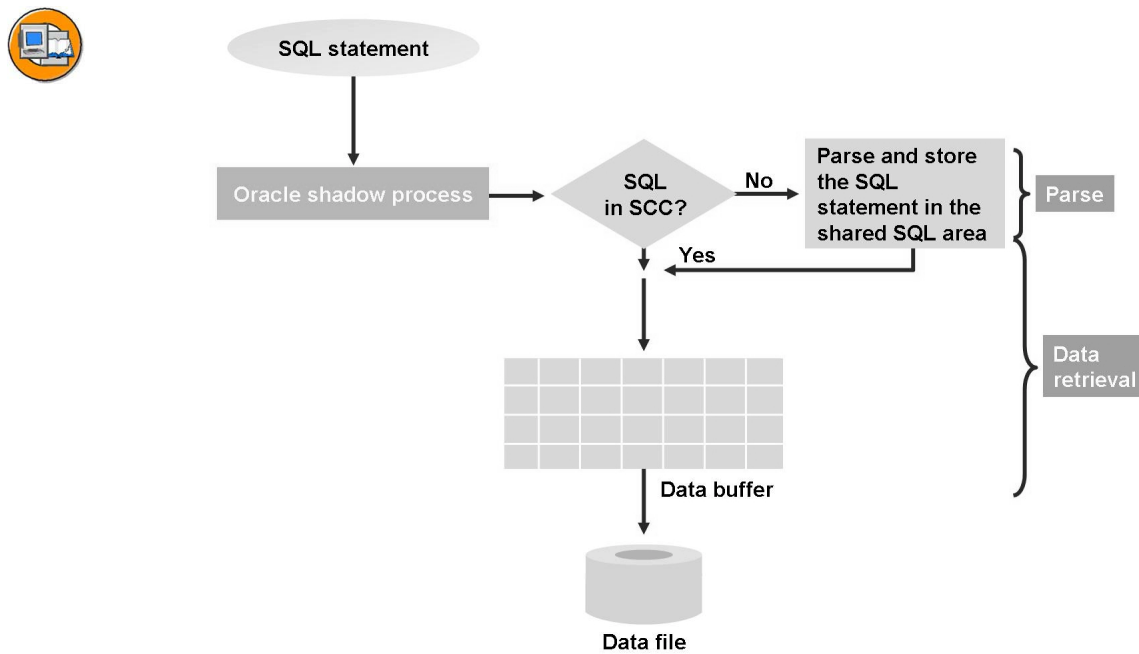


Figure 131: How Oracle Processes an SQL Statement

Logical Sequence of Database Operations

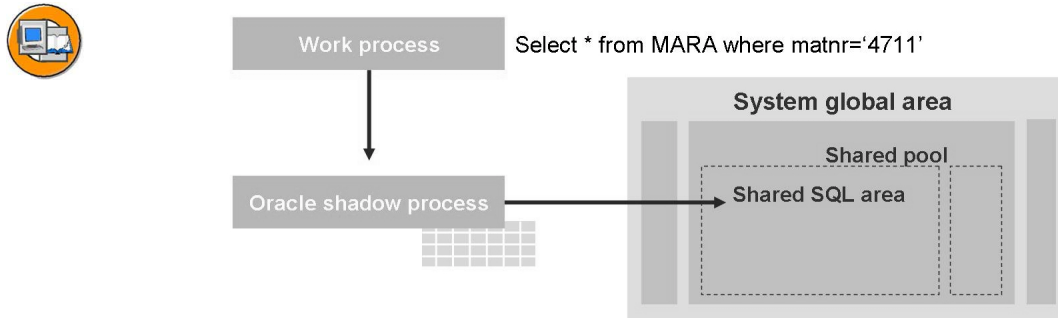
Database requests are interconnected and always occur in the same logical sequence.

- The DECLARE function defines and numbers the cursor. DECLARE precedes the PREPARE function.
- The PREPARE function prepares a specific SQL statement, such as: `Select * from MARA where matnr='4711'`, and defines the execution plan before the system can transfer the request to the database. During this preparation, the system is concerned only with the structure of the SQL statement and not with the values it contains. To share the execution plans for a large number of statements, the SAP system uses bind variables rather than literals in SQL statements:
`SELECT * FROM MARA WHERE MANDT=:A0 AND MATNR=:A1.`
- The OPEN function takes the prepared SELECT statement and completes it with the correct values: `SELECT * FROM MARA WHERE MANDT='100' AND MATNR='4711'`. In the above example, OPEN would issue the value **4711** to the field *matnr*.
- FETCH passes the entries from the database to the database interface of the SAP system. All of the database operations required to execute an SQL statement are linked by the same cursor ID.

If the SQL statement makes changes in the database (INSERT, UPDATE, DELETE), PREPARE is followed by EXEC, which executes the statement.

A single SQL statement may generate multiple fetch operations, depending on the length of the row of data and the number of rows retrieved.

If the system can refer back to an SQL statement that has already been prepared, there is no PREPARE operation, and the statement is executed using REOPEN or REEXEC, as appropriate.



- 1) **DECLARE** SELECT * FROM MARA WHERE MANDT=:A0 AND MATNR=:A1
Shadow process allocates memory areas from the Program Global Area (PGA)
- 2) **PREPARE**
Determination of the execution plan
- 3) **OPEN** SELECT * FROM MARA WHERE MANDT='100' AND MATNR='4711'
Substitution of the variables and execution of the statement
- 4) **FETCH**
Fetch the results

Figure 132: Logical Sequence of Database Operations

Exercise 14: Activation of Dynamic System Global Area

Exercise Objectives

After completing this exercise, you will be able to:

- Activate dynamic System Global Area

Business Example

After an upgrade to Oracle 9.2, you want to activate Oracle dynamic System Global Area.

Task 1:

Make sure that the Oracle server parameter file is activated on your play database.

1. Check which parameter file is used by your play database (`init<DBSID>.ora` or `spfile<DBSID>.ora`).
2. Activate the Oracle server parameter file, if it is not already activated.

Task 2:

Check if the dynamic System Global Area is activated.

1. Check the values of the following parameters:
 `DB_BLOCK_BUFFERS`.
 `DB_CACHE_SIZE`.

Task 3:

Activate the dynamic System Global Area.

1. Use BR*Tools to reset the parameter `DB_BLOCK_BUFFERS`.
2. Estimate good start values for `SGA_MAX_SIZE` and `DB_CACHE_SIZE`.
3. Use BR*Tools to change the parameter value of `DB_CACHE_SIZE` and `SGA_MAX_SIZE`.
4. Restart the database to activate your settings.

Solution 14: Activation of Dynamic System Global Area

Task 1:

Make sure that the Oracle server parameter file is activated on your play database.

1. Check which parameter file is used by your play database (init<DBSID>.ora or spfile<DBSID>.ora).

- a) Use BR*Tools or the SQL*Plus command `show parameter spfile` to check which parameter file is used by your play database.

If init<DBSID>.ora is used, the field *VALUE* is empty:

```
SQL> show parameter spfile
```

NAME	TYPE	VALUE

spfile	string	

Else if spfile is used, the field *VALUE* contains the name and path of the spfile:

```
SQL> show parameter spfile
```

NAME	TYPE	VALUE

spfile	string	%ORACLE_HOME%\DATABASE\SPFILE%ORACLE_SID%.ORA

2. Activate the Oracle server parameter file, if it is not already activated.
 - a) Use SQL*Plus to create the spfile.

```
SQL> create spfile from pfile;
```
 - b) Do not forget to restart the database.

Task 2:

Check if the dynamic System Global Area is activated.

1. Check the values of the following parameters:
 DB_BLOCK_BUFFERS.

Continued on next page

DB_CACHE_SIZE.

- a) Use BR*Tools or SQL*Plus to check the values of these parameters. DB_BLOCK_BUFFERS is set to 2048 and DB_CACHE_SIZE is set to 0, so dynamic SGA is not activated on the play database.

Task 3:

Activate the dynamic System Global Area.

1. Use BR*Tools to reset the parameter DB_BLOCK_BUFFERS.
 - a) Use BRGUI or BRTOOLS and chose *Instance Management* → *Alter database parameters*.
 - b) In the next menu, *BRSPACE options for alter database parameters*, enter the database parameter DB_BLOCK_BUFFERS and choose *Continue*.
 - c) In the following menu, *Alter database parameter main menu*, choose *Reset parameter value*.
 - d) Use the default settings and choose *Continue*. Make sure that the value in the *New parameter value (value)* field is set to **null**.
 - e) Choose *Exit program*.
2. Estimate good start values for SGA_MAX_SIZE and DB_CACHE_SIZE .
 - a) A good value for DB_CACHE_SIZE is the value of the old parameter, DB_BLOCK_BUFFERS.



Caution: These two parameters have different units. DB_CACHE_SIZE is given in bytes and DB_BLOCK_BUFFERS in blocks.

- b) Use the default value for SGA_MAX_SIZE.

Continued on next page

3. Use BR*Tools to change the parameter value of DB_CACHE_SIZE and SGA_MAX_SIZE.
 - a) Use BRGUI or BRTOOLS and chose *Instance Management* → *Alter database parameters*.
 - b) In the next menu, *BRSPACE options for alter database parameters*, enter the database parameter DB_CACHE_SIZE and choose *Continue*.
 - c) In the following menu, *Alter database parameter main menu*, choose *Change parameter value*.
 - d) In the following menu *Options for alter database parameter*, enter the calculated new parameter value and select *scope = spfile*. Then choose *Continue*.
 - e) Choose *Exit program*.
 - f) Go back to the menu, *BRSPACE options for alter database parameters*, enter the database parameter SGA_MAX_SIZE, and choose *Continue*.
 - g) In the following menu, *Alter database parameter main menu*, choose *Change parameter value*.
 - h) In the following menu, *Options for alter database parameter* enter the old parameter value in the field new parameter value and select *scope = spfile*. Then choose *Continue*.
 - i) Choose *Exit program*.
4. Restart the database to activate your settings.



Lesson Summary

You should now be able to:

- Explain the concepts of Oracle cache management
- Describe the components and the architecture of the System Global Area
- Describe Oracle's dynamic storage management, known as the dynamic System Global Area
- Discuss how Oracle processes an SQL statement

Lesson: Oracle Program Global Area

Lesson Overview

This lesson introduces the Oracle Program Global Area, including the automatic Program Global Area.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe the advantages and the features of the automatic Program Global Area

Business Example

Administrators of SAP NetWeaver BI databases should pay attention to the tuning of the Program Global Area due to memory-intensive SQL operations typical to the data warehouse area (large sorts and joins). The automatic Program Global Area is particularly suited to and beneficial when using with SAP NetWeaver BI applications. However, it can also be beneficial when used with other non-BI SAP applications.

Introduction to PGA Memory Management

The Program Global Area (PGA), often known as the Process Global Area or work area, resides in the private memory of the shadow process. It contains global variables, data structures, and control information for each shadow process.

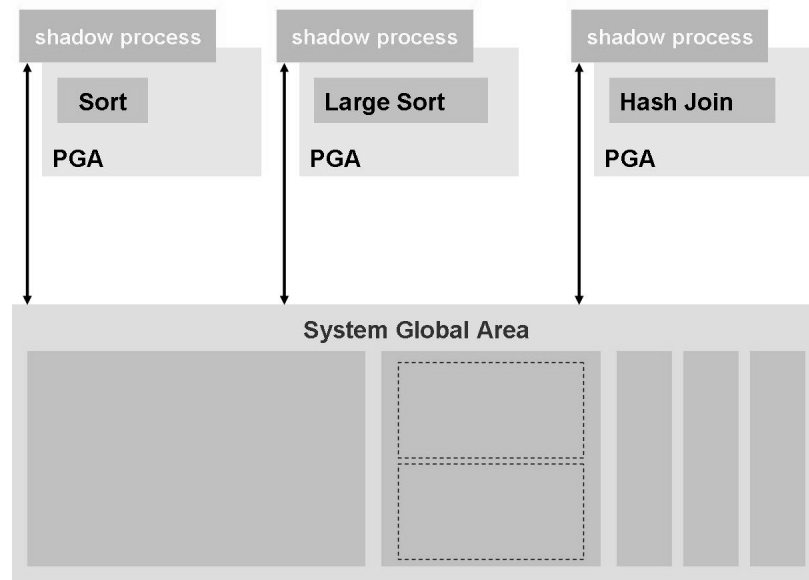


Figure 133: Program Global Area

The size of a work area can be controlled and tuned. Generally, bigger work areas can significantly improve the performance of a particular operator at the cost of higher memory consumption. Ideally, the size of a work area is big enough that it can accommodate the input data and auxiliary memory structures allocated by its associated SQL operator (for example, a memory sort). This is known as the optimal size of a work area.

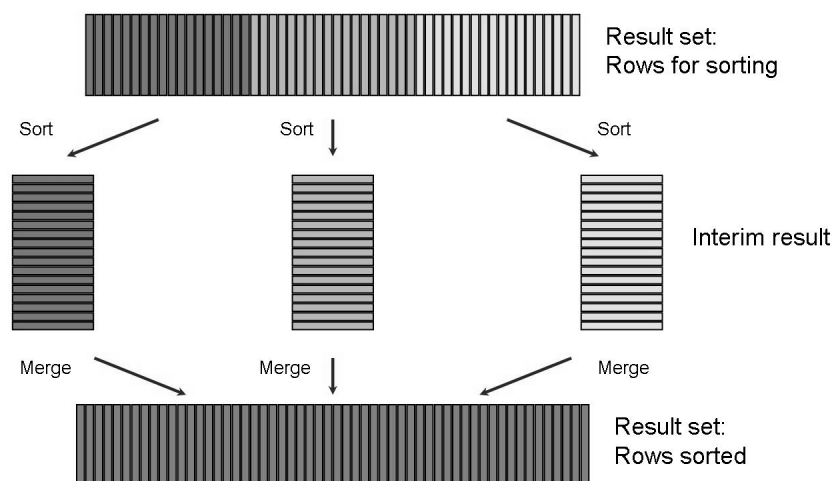
The PGA consists of an area that can be optimized (tunable) and an area which cannot be optimized (non-tunable). Since the speed (performance) of queries is also largely dependent on the amount of available PGA memory, you should pay attention to the tuning of this memory.

Performance usually increases considerably if more memory is available. Ideally, the work area should be just large enough to contain all the data and auxiliary structures to execute the current SQL operation. This ideal size is then called the **optimal work area size**. The execution is then included in the statistics under *optimal_executions*. Making the work area too large does not improve performance; the allocated memory is therefore occupied in vain.

However, if the work area is too small, the system has to use the disk as a temporary storage area (temporary tablespace) for the sort operation, which causes the response time to increase significantly due to the additional I/O. If a once-off transfer of an interim result to the disk is sufficient, this is referred to as a **one-pass** sort and the size of the work areas is referred to as a one-pass work area size.

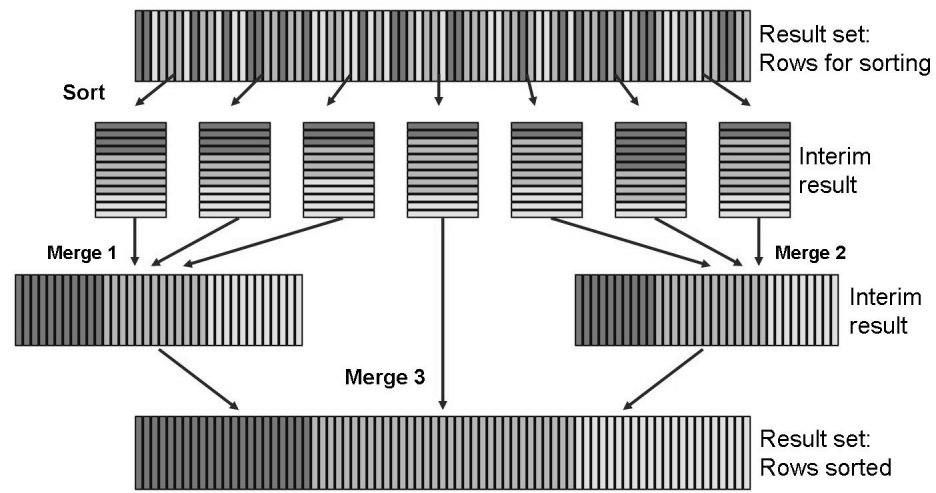
If several additional passes are necessary to execute the SQL operation, the size of the work area is referred to as a **multipass** work area size. In this case, the response time increases drastically compared with the optimum time, so this should be avoided. The aim here is to have most work areas in the optimum range so that most queries are processed in the optimum response time (for example, 90% in the case of OLTP). A small number of queries can then appear in the one-pass area (for example, less than 10% in the case of OLTP). You should avoid the multipass area if possible.

Work Area Terminology



The data has been read, sorted, and dumped to disc in chunks, then re-read once to be merged into order, and dumped again.

Figure 134: One-Pass Sort



After sorting the data in chunks, Oracle was unable to re-read the top of every chunk simultaneously, so there are multiple merge passes.

Figure 135: Multipass Sort

PGA Management in Oracle 8i

Up to Oracle 8i, the database administrator was responsible for setting up the PGA accurately and optimally. This occurred by specifying the following Oracle parameters:



- SORT_AREA_SIZE
- HASH_AREA_SIZE
- BITMAP_MERGE_AREA_SIZE
- CREATE_BITMAP_AREA_SIZE

These PGA parameters specify the allocated PGA memory of an Oracle shadow process in manual PGA management mode.



`SORT_AREA_SIZE` is set to 10MB
`HASH_AREA_SIZE` is set to 20MB

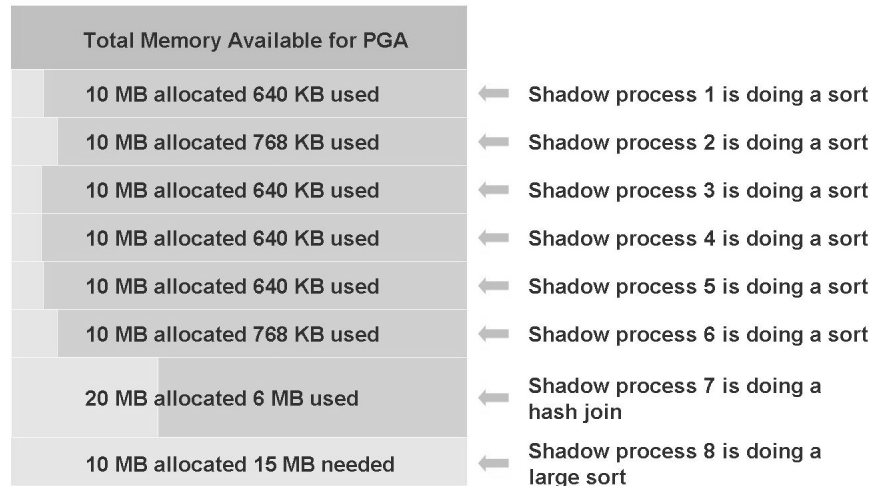


Figure 136: “Old” PGA Management

The PGA management of Oracle 8i has some issues. The most problematic aspect of the previous approach was that the values were the same size for all shadow processes (see figure above). Furthermore, you selected these sizes depending on the anticipated dataset as well as on the total number of work areas, that is, the number of shadow processes. Both sizes can vary so that it is difficult to find the best configuration in each case. With Windows NT/2000, there was the additional restriction of the maximum amount of memory that could be allocated and the fact that the PGA memory was part of the SGA memory (`ORA-04030: out of memory`).

PGA Management in Oracle 9i or Higher

With Oracle 9i, it is possible to leave this PGA tuning task to the Oracle server, which makes the database easier to administer (and more efficient). As of Oracle 9i, you can use this feature to adjust the PGA automatically and dynamically, thereby avoiding problems with too little or too much allocated PGA memory.



WORKAREA_SIZE_POLICY is set to AUTO
PGA_AGGREGATE_TARGET is set to 90MB

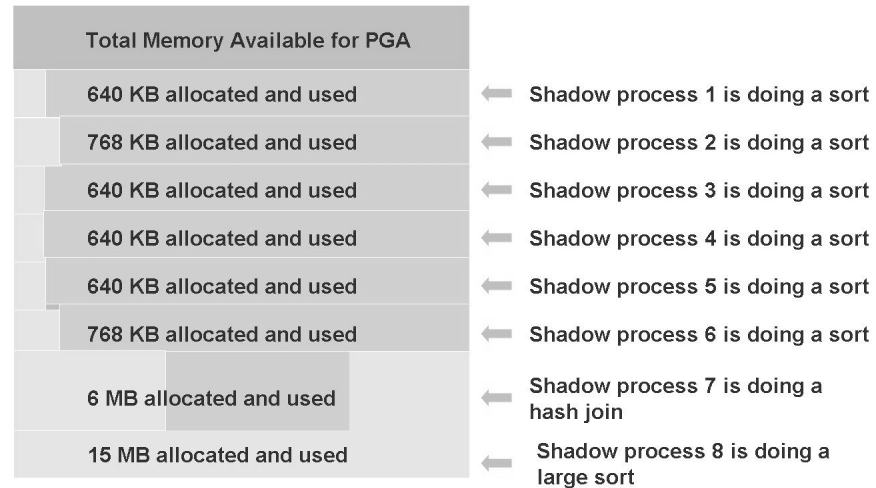


Figure 137: Automatic PGA Memory Management

The database administrator now only has to determine the maximum total PGA memory of an instance (PGA_AGGREGATE_TARGET). To maximize overall performance through efficient memory usage, the Oracle server uses an intelligent algorithm to divide this memory between the existing Oracle server processes depending on demand.

The size of the total PGA memory can be changed at the instance runtime, that is, dynamically. Since the Oracle server is most familiar with the SQL commands currently being executed, we recommend that you use this feature instead of trying to configure the PGA manually. Doing this results in a higher throughput, as well as a shorter response time.

Advantages of Automatic PGA

Automatic PGA provides following advantages:



- Simple PGA memory tuning is available.
- The database administrator's work is reduced.
- The total PGA memory is easier to quantify than the sort area per dedicated database shadow process.
- No recalculation of the SORT_AREA_SIZE is necessary if the number of work processes increases.
- Better memory use: Physical memory is only allocated by the shadow process that actually needs the memory for a sort or a similar process.
- A shadow process returns allocated PGA memory that is no longer required to the Oracle PGA management so that another shadow process can use this memory.
- There are options to achieve efficient monitoring, tuning, and simulation within the Oracle server.
- Higher system performance, shorter response times, and more efficient memory usage: The ideal situation, in which the optimum amount of PGA memory is available to the Oracle shadow process, means that SQL queries are processed faster. The response time (OLTP) decreases or the system performance (OLAP, DSS) increases.

In particular, the fact that a shadow process can release allocated PGA memory again (that is, return it to the operating system) so that another shadow process can use this memory is important for SAP systems because, in the SAP system, a dedicated Oracle shadow process is tied to an SAP work process, usually for the duration of the work process. In the case of manual PGA management, a shadow process only returns the PGA memory allocated at the start of the process to the operating system once the process has ended. Using automatic PGA, the allocated memory is used more efficiently.

However, one possible disadvantage is that too much PGA memory may be reserved in advance and consequently wasted if the calculated PGA_AGGREGATE_TARGET is too high. To prevent this, you should use the monitoring options, which enable you to recognize an over-allocation of PGA memory.

Functions and Restrictions of Automatic PGA

The automatic PGA is activated using the following parameters:



WORKAREA_SIZE_POLICY

Using this parameter, you can set the automatic PGA tuning dynamically to AUTO or switched off (MANUAL).

Oracle Release 10g – Default = AUTO

Oracle Release 9i – Default = MANUAL

PGA_AGGREGATE_TARGET

This parameter defines the maximum amount of memory – in bytes, Kilo (K), Mega (M), or Giga (G) – that can be used for PGA. It is dynamically adjustable.

To activate automatic PGA tuning, parameters PGA_AGGREGATE_TARGET and WORKAREA_SIZE_POLICY=AUTO must both be set, or else the conventional PGA storage management is switched on. For PGA_AGGREGATE_TARGET, the following value range applies: 10M < PGA_AGGREGATE_TARGET < 4096G -1

The old PGA-parameters SORT_AREA_SIZE, HASH_AREA_SIZE, BITMAP_MERGE_AREA_SIZE, CREATE_BITMAP_AREA_SIZE, and SORT_AREA_RETAINED_SIZE are ignored by Oracle if automatic PGA tuning is active.

Memory Allocation

PGA_AGGREGATE_TARGET limits both the globally available PGA memory for the work areas of all Oracle shadow processes and (using internal Oracle limits) the work area of an individual shadow process. This prevents a single, very large sort process from using up too much memory.



Caution: Since the Oracle shadow processes allocate and use this memory, and since these processes run on the database server, you must take the memory requirement of the PGA into account when calculating the available memory for the SGA (SGA_MAX_SIZE).

When calculating an access plan, the Oracle cost-based optimizer takes the PGA_AGGREGATE_TARGET parameter into account. According to this parameter the minimum and maximum amounts of available main memory are included into the cost calculation for a later execution time. Switching on this feature may therefore lead to changes in the access plans.

The allocation of the total PGA memory occurs according to an Oracle algorithm, which takes the following into account:



- The total PGA size of all server processes cannot exceed the value of `PGA_AGGREGATE_TARGET`.
- The system tries to allocate each process with as much memory as it requires.
- Throughput and response times should be optimized equally. Memory-intensive operations are given priority when memory is allocated.

Over and above these general aims, the algorithm takes account of some additional limits that ensure that the system does not allocate the total PGA memory to a single shadow process. These limits are:



- A maximum of 5% of the `PGA_AGGREGATE_TARGET` for a shadow process
- A maximum of 30% of the `PGA_AGGREGATE_TARGET` for a shadow process in the case of parallel operations
- A maximum of 200 MB for a shadow process



Caution: In theory, these limits may change if you import a patch or a patch set, or if you use a new Oracle version.

Enabling Automatic PGA

When implementing this feature, you should take particular care that the sort areas (specified by the `SORT_AREA_SIZE` and other PGA parameters) reserved up to now for each process do not become smaller. This would cause sorts that were previously executed in the memory to be executed on disk, which would make them considerably slower.

To specify `PGA_AGGREGATE_TARGET`, observe the following general procedure:



1. Calculate or estimate (empirically) a good start value.
2. Monitor `PGA_AGGREGATE_TARGET` (under a representative system workload).
3. Optimize `PGA_AGGREGATE_TARGET`.

Calculate or Estimate a Good Start Value

The following general recommendations apply to a standalone database server.

- For OLTP systems:

$$\text{PGA_AGGREGATE_TARGET} = \text{<Total physical memory>} * 20\%$$

- For data warehouse systems:

$$\text{PGA_AGGREGATE_TARGET} = \text{<Total physical memory>} * 40\%$$

- Use the following formula as an additional guide value and lower limit:

$$\text{pga_aggregate_target} \geq (\text{SORT_AREA_SIZE} + \text{HASH_AREA_SIZE} + \dots) *$$

This formula is based on the assumption that not all shadow processes are simultaneously busy with larger sort tasks. Therefore, only the number of shadow processes divided by 10 should be used to calculate an approximation for PGA_AGGREGATE_TARGET. You can use the Oracle profile parameter processes to determine the number of shadow processes. This parameter determines the maximum number of Oracle shadow processes.

Since the database server must have sufficient physical memory for the SGA, for the PGA and, if necessary, for other non-Oracle processes, PGA_AGGREGATE_TARGET is also restricted by the following formula:
$$\text{SGA_MAX_SIZE} + \text{PGA_AGGREGATE_TARGET} \leq \text{<phys. memory of DB server>}$$

The implicit restriction that applies to this formula is that no paging should take place during the Oracle processes.

For further information on Oracle recommendations for OLTP applications and Oracle recommendation for data warehouse applications, please check SAP Note 124361 and SAP Note 632556, respectively.

Exercise 15: Activation of Automatic Program Global Area

Exercise Objectives

After completing this exercise, you will be able to:

- Check whether automatic Program Global Area is activated
- Activate automatic Program Global Area

Business Example

After an upgrade to Oracle 10g, you want to check whether the Oracle automatic Program Global Area is activated.

Task 1:

Check if “old” PGA management or automatic PGA management is activated in your play database.

1. Check if the “old” PGA management or the automatic PGA management is activated in your play database.

Task 2:

Set the value for PGA_AGGREGATE_TARGET to its minimal value.

1. The minimal value you can use for PGA_AGGREGATE_TARGET is 10 MB.
2. Use BR*Tools to change the parameter value of PGA_AGGREGATE_TARGET.

Solution 15: Activation of Automatic Program Global Area

Task 1:

Check if “old” PGA management or automatic PGA management is activated in your play database.

1. Check if the “old” PGA management or the automatic PGA management is activated in your play database.
 - a) Use BR*Tools or the SQL*Plus command `show parameter <name of the parameter>` to check the value of parameter `WORKAREA_SIZE_POLICY`. If this parameter is set to **MANUAL**, the old PGA management or automatic PGA management is activated.

Task 2:

Set the value for `PGA_AGGREGATE_TARGET` to its minimal value.

1. The minimal value you can use for `PGA_AGGREGATE_TARGET` is 10 MB.
2. Use BR*Tools to change the parameter value of `PGA_AGGREGATE_TARGET`.
 - a) Use BRGUI or BRTOOLS and chose *Instance Management* → *Alter database parameters*.
 - b) In the next menu, *BRSPACE options for alter database parameters*, enter the database parameter `PGA_AGGREGATE_TARGET` and choose *Continue*.
 - c) In the following menu, *Alter database parameter main menu*, choose *Change parameter value*.
 - d) In the following menu, *Options for alter database parameter*, enter the new parameter value (**10M**), select **scope = both**, and choose *Continue*.



Lesson Summary

You should now be able to:

- Describe the advantages and the features of the automatic Program Global Area



Unit Summary

You should now be able to:

- Explain the concepts of Oracle cache management
- Describe the components and the architecture of the System Global Area
- Describe Oracle's dynamic storage management, known as the dynamic System Global Area
- Discuss how Oracle processes an SQL statement
- Describe the advantages and the features of the automatic Program Global Area



Test Your Knowledge

1. Which memory areas are part of the Oracle System Global Area?
Choose the correct answer(s).
 - ☐ A Redo buffer
 - ☐ B Sort cache
 - ☐ C Data cache
 - ☐ D Shared SQL area

2. Which parameters can be changed dynamically when using dynamic System Global Area?
Choose the correct answer(s).
 - ☐ A SGA_MAX_SIZE
 - ☐ B DB_CACHE_SIZE
 - ☐ C DB_BLOCK_BUFFERS
 - ☐ D SHARED_POOL_SIZE
 - ☐ E BLOCK_SIZE

3. The shared SQL area is a memory structure that contains the parsed SQL statements and execution plans.
Determine whether this statement is true or false.
 - ☐ True
 - ☐ False

4. Which of the following parameters are used to activate automatic Program Global Area?
Choose the correct answer(s).
 - ☐ A PGA_AUTO_ENABLE
 - ☐ B WORKAREA_SIZE_POLICY
 - ☐ C PGA_AGGREGATE_TARGET
 - ☐ D PGA_AREA_SIZE



Answers

1. Which memory areas are part of the Oracle System Global Area?

Answer: A, C, D

The System Global Area contains shared memory areas that can be called by all Oracle processes.

2. Which parameters can be changed dynamically when using dynamic System Global Area?

Answer: B, D

Using dynamic SGA, the parameters `DB_CACHE_SIZE` and `SHARED_POOL_SIZE` can be changed dynamically.

3. The shared SQL area is a memory structure that contains the parsed SQL statements and execution plans.

Answer: True

4. Which of the following parameters are used to activate automatic Program Global Area?

Answer: B, C

Automatic PGA is activated using the parameters `WORKAREA_SIZE_POLICY` and `PGA_AGGREGATE_TARGET`.

Unit 6

Monitoring of the database instance

Unit Overview

This unit introduces the features and the usage of the performance monitor as part of the DBA Cockpit.



Unit Objectives

After completing this unit, you will be able to:

- Use the DBA Cockpit to analyze the performance of the Oracle database
- Describe how the DBA Cockpit supports the database features of Oracle 9i and Oracle 10g
- Use the enhanced history functionality of the DBA Cockpit

Unit Contents

Lesson: Monitoring Performance Using the DBA Cockpit.....	404
Exercise 16: Navigation in the DBA Cockpit Performance Monitor.....	429

Lesson: Monitoring Performance Using the DBA Cockpit

Lesson Overview

This lesson introduces performance monitoring as part of the new DBA Cockpit (transaction DBACOCKPIT).



Lesson Objectives

After completing this lesson, you will be able to:

- Use the DBA Cockpit to analyze the performance of the Oracle database
- Describe how the DBA Cockpit supports the database features of Oracle 9i and Oracle 10g
- Use the enhanced history functionality of the DBA Cockpit

Business Example

You want to use the new database features of Oracle 9i and Oracle 10g, for example, dynamic SGA management and dynamic PGA management. Therefore, you need to monitor these new features.

Introduction

The new DBA Cockpit, transaction DBACOCKPIT, can be used to analyze single Oracle database instances as well as Oracle real application cluster (RAC) installations. Transaction DBACOCKPIT is the successor of the old database monitors, such as transaction ST04 and the planning calendar (transaction DB13). The new database monitor retrieves information from Oracle performance views V\$, GV\$, and DBA.



Hint: The global views, GV\$, are only available with Oracle RAC installations.

The performance monitor gives a general overview of database performance. It provides different ways of looking at the monitoring information:



- A main monitor with an overview of the database performance
- Several sub-monitors for detailed performance analyses, grouped as follows:
 - Wait event analysis
 - SQL statement analysis
 - Statistical information
 - Feature monitoring
 - Additional functions
 - RAC statistics



Caution: As of SAP NetWeaver 7.00 SP12, DBACOCKPIT is the new central transaction for database administration. To monitor the database performance in former releases of the SAP Web Application Server you can use the old database monitor, ST04, which was available for all database types supported by SAP. For SAP Web Application Server 6.40, the performance monitor was reengineered to monitor the new features of Oracle 9i. The new monitoring transaction, ST04N, is only applicable to Oracle 9i-based systems. Transaction ST04N is presented in the appendix of this lesson.

Prerequisites

The prerequisites described here are required for monitoring and administration of the local system (that is, the system on which the DBA Cockpit is running).

- Some performance monitors within the DBA Cockpit require special database objects. These objects are created using an SQL script. See SAP Note 706927 for this script and more information.
- The new planning calendar requires BR*Tools 700, patch level 24 or higher.
- Some functions in the DBA Cockpit require the Oracle Active Workload Repository. This Oracle option must therefore be available for selection (see SAP Note 1028068).



Hint: For further details about the prerequisites of the DBA Cockpit see SAP Note 1028624.

To have a performance history, ABAP report **RSORAHCL** must be scheduled as a periodic batch job via transaction SM36. The scheduling details are stored in table TCOLL. To edit the scheduling data, call either ABAP report **RSCOLL40** or transaction ST03N.

The Performance Overview Monitor

The performance overview monitor, which is the entry screen of the performance monitor in the new DBA Cockpit, provides structural and efficiency data on the following items:

- General information
- Data buffer
- Shared pool
- Log buffer
- Calls
- Time statistics
- Redo logging
- Table scans and fetches
- Sorts
- Instance efficiency

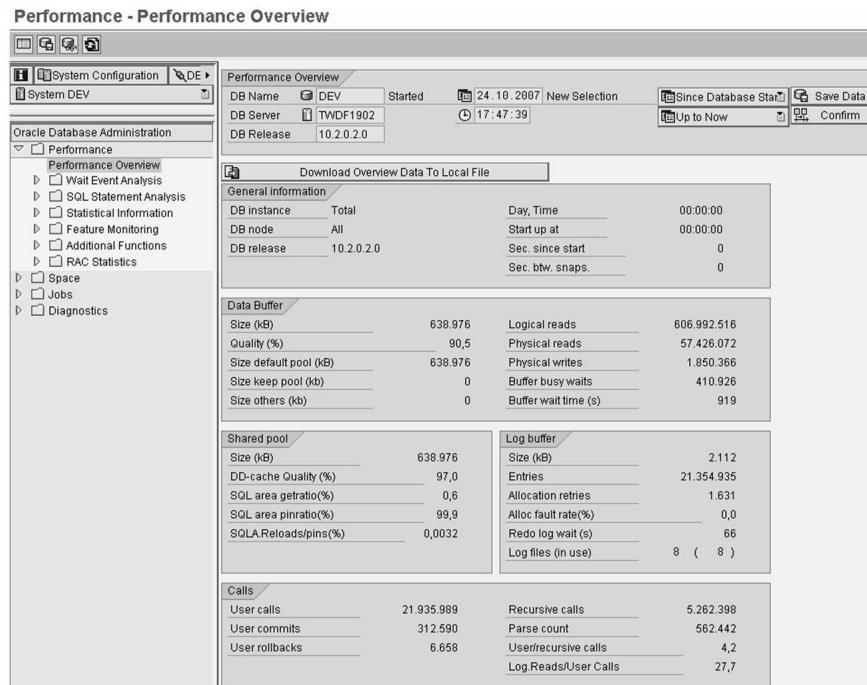


Figure 138: The Performance Overview Monitor

Many numbers depend on each other. The following checklist names a few key performance figures of your Oracle database.

- The **data buffer quality** is based on the ratio of physical reads to the total number of reads. The lower the ratio, the better the buffer quality. The data buffer quality should be better than 94,0%; the statistics should be based on 15 million total reads. This number of reads ensures that the database is in a state of equilibrium.
- Good performance is indicated by a **ratio of user and recursive calls** that is greater than 2. Otherwise, the number of recursive calls compared to the number of user calls is too high. Over time, this ratio always declines because more and more SQL statements are parsed in the meantime.
- If the **number of reads per user call** exceeds 30 blocks per user call, this might indicate an expensive SQL statement.
- Check the value of **time/user call**. Values larger than 15 ms often indicate an optimization issue.
- Compare **busy wait time versus CPU time**. A ratio of 60:40 generally indicates a well-tuned system. Significantly higher values (for example, 80:20) indicate room for improvement.
- The **DD cache quality** should be better than 80%.

In addition to the performance overview monitor, transaction DBACOCKPIT provides specialized sub-monitors for more detailed analysis.

Enhanced History Functions

Some sub-monitors provide history data, called snapshots. You even can select the time frame that should be displayed. By default, the time span is from DB start up to now. The following table summarizes all possible options.



Selected History Options

Since	Up to	Remark
DB start	Now	Displays the changes from database start to the current time
DB start	snapshot	Displays the changes from database start to the selected snapshot time
snapshot	Now	Displays the changes from your selected snapshot to the current time
snapshot1	snapshot2	Displays the changes between your selected snapshots

To choose another time frame, double-click the corresponding delimiter in the *Selected History*.

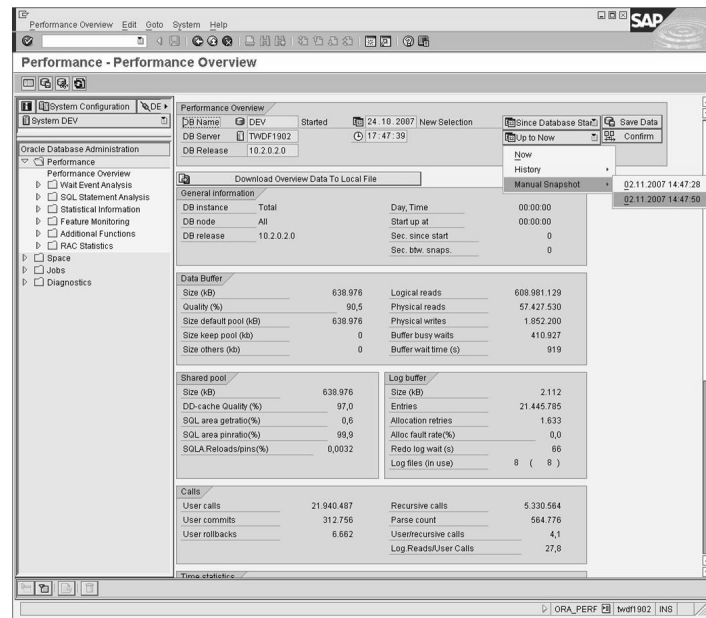


Figure 139: Selecting the History Time Frame for Enhanced History Function

Enhanced History Function

As mentioned earlier, some sub-monitors provide history data. The history is stored in corresponding SAP tables. Oracle performance views, which are displayed in lowercase letters, have such corresponding tables. The tables are updated by ABAP report **RSORAHCL** (or **RSORAHIST**). History data of global views is stored in SAP tables **GVD_***. Note that the Oracle history data is not stored in SAP table **MONI**, which contains the history of SAP performance data.

The Sub-Monitors in Detail

The following section gives a short overview of the performances monitors in the DBA Cockpit.

Wait Event Analysis

The Wait Event Analysis monitor collects the following sub-monitors:

- Session Monitor
- Buffer Busy Wait
- System Event
- System Event History
- Filesystem Requests
- Enqueue Stat
- Lock Monitor
- Latch Monitor
- Active Session History
- Workload Reporting



Analyze DB Performance: Oracle Session

Refresh Set Selection Criteria...

System Configuration DE System DEV

Oracle Database Administration

- Performance
 - Performance Overview
 - Wait Event Analysis
 - Session Monitor
 - Buffer Busy Wait
 - System Event
 - System Event History
 - Filesystem Requests
 - Enqueue Stat
 - Lock Monitor
 - Latch Monitor
 - Active Session History
 - Workload Reporting
 - SQL Statement Analysis
 - Statistical Information
 - Feature Monitoring
 - Additional Functions
 - RAC Statistics
 - Space
 - Jobs
 - Diagnostics

Session Monitor

DB Name: DEV Started: 24.10.2007
DB Server: TWDF1902 17:47:39
DB Release: 10.2.0.2.0

Analyze since DB start.

* S	Oracle	Client syst	Client	Status	Event	SQL Statement	Logical R	Phys. Re.	Block Ch.
86	6468	TMPWORKIT	7540.76	INACTIVE	SQL*Net mes		42976	6773	2802
88	4280	TMPWORKIT	5052.68	INACTIVE	SQL*Net mes		36	0	0
89	464	TMPWORKIT	6572.64	INACTIVE	SQL*Net mes		82300	27172	7709
93	4776	TMPWORKIT	7712.54	INACTIVE	SQL*Net mes		53446	913	7753
95	6680	TMPWORKIT	4128.75	INACTIVE	SQL*Net mes	SELECT TABNAME, BLOCKNR, I91	0	0	0
97	7316	TMPWORKIT	6980.81	INACTIVE	SQL*Net mes		53568	11951	1423
98	4988	TMPWORKIT	6696.62	INACTIVE	SQL*Net mes		5134188	304708	9337
101	5980	TWDF1902	Shadow	ACTIVE	Streams AQ:		0	0	0
102	7556	TMPWORKIT	6884.41	INACTIVE	SQL*Net mes		52701	567	7498
104	7272	TMPWORKIT	3696.75	INACTIVE	SQL*Net mes		5438	137	2004
105	6504	TMPWORKIT	8072.53	INACTIVE	SQL*Net mes		3289	111	1134
106	6532	TMPWORKIT	4468.69	INACTIVE	SQL*Net mes		86056	604	8822
107	6316	TMPWORKIT	6312.65	INACTIVE	SQL*Net mes		9384	367	4034
108	5972	TMPWORKIT	4252.53	INACTIVE	SQL*Net mes		54153	286	8184
110	4316	TMPWORKIT	1616.64	ACTIVE	SQL*Net mes	SELECT T1.INST_ID, T1.SID, T1	9439500	90116	339259
111	6628	TMPWORKIT	440.490	INACTIVE	SQL*Net mes		49614	271	7577

Figure 140: Wait Event Analysis Sub-Monitors

These sub-monitors in the DBA Cockpit help you analyze SQL statements.

The following table summarizes the sub-monitors, their data sources and descriptions.

Wait Event Analysis Sub-Monitors

Sub-Monitor	Data Source	Remarks
Session Monitor	several views, for example, V\$Session, V\$Process, and so on	This monitors the Oracle session list and related resource information. In addition, you can see an execution plan and the SQL statement performed by a session.
Buffer Busy Waits		A buffer busy wait indicates that there are some buffers in the buffer cache that multiple processes are attempting to access concurrently.
System Event	GV\$SYSTEM_EVENT	This sub-monitor lets you check wait events and system events in the Oracle database.
System Event History		Gives you an overview of the history of system events of the Oracle instance(s).
Filesystem Requests	GV\$FILESTAT	Data file activity has an effect on database performance. This view helps you to identify the frequently used data files and put them on separate disks to avoid contention, if necessary.
Enqueue Stat	V\$ENQUEUE_STAT	This monitor helps you to monitor enqueues.
Lock Monitor	V\$LOCK* views	This monitors currently active locks.
Latch Monitor	several views, e.g., V\$SQL, V\$SYSSTAT and GV\$LATCH*	This monitors the latch activity. Processes have to obtain a latch in order to access the data.

Undo Statistics	GV\$UNDOSTAT	This sub-monitor lets you check the undo statistics.
Active Session History	GV\$ACTIVE_SESSION_HISTORY	This shows the Oracle active session history views, which provide sampled session activity in the database instances.
Workload Reporting	AWR	This sub-monitor displays very detailed reports from the Oracle Active Workload Repository.

SQL Statement Analysis

The SQL Statement Analysis monitor collects the following sub-monitors:

- Shared Cursor Cache
- Table Access
- Column Usage
- System Statistics for CBO



Analyze DB Performance: Shared Cursor Cache

<input type="checkbox"/> Refresh <input type="button" value="Set Selection Criteria..."/>			
Shared Cursor Cache DB Name: DEV Started: 24.10.2007 DB Server: TWDF1902 17:47:39 DB Release: 10.2.0.2.0			
<input type="button" value="EXPLAIN"/> <input type="button" value="ABAP Source"/> <input type="button" value="Table f"/>			
Analyze since DB start.			
Elapsed Time	Elapsed Time/Exec	SQL statement	CPU Time
6.521.276.276	815.159.534,5	call dbms_space.auto_space_advisor_job_proc ()	670.984.559
5.383.671.353	9.860.203,9	SELECT OWNER, SEGMENT_NAME, PARTITION_NAME, SEGMENT_TYPE, TA	643.914.800
2.105.145.212	11.826.658,5	select count (*) from dba_segments where max_extents >= ex	405.477.483
186.260.669	980.319,3	select tablespace_name, next_extent, count (*) from dba_se	159.539.826
1.787.584.479	11.607.691,4	select min (max_extents - extents) from dba_segments where	351.472.557
1.705.069.402	11.759.099,3	select max (extents) from dba_segments	312.263.410
121.460.390	943.474,9	select count (*) from dba_segments where owner = 'A0' and	120.992.619
430.399.650	3.586.663,8	select index_name, table_name, uniqueness from user_indexes	122.447.042
16.726.426	278.773,8	select view_name from user_views order by view_name	14.735.563
142.937.425	23.822.904,2	select a.username user_name, a.user_id user_id,	37.220.420
241.103.327	272,7	INSERT INTO SNAP (DATUM, UZEIT, AHOST, UNAME, MANDT, MODNO,	159.661.439
219.111.123	12.172.840,2	SELECT OWNER, SEGMENT_NAME, SEGMENT_TYPE, TABLESPACE_NAME, H	52.867.561
699.067.535	1.360,3	INSERT INTO "VBDATA" VALUES('A0', 'A1', 'A2', 'A3', 'A4')	390.213.560
206.796.698	34.466.116,3	select a.segment_type, round(a.mb,2) size_mb,	36.299.712
35.688.084	159,7	SELECT * FROM "VBKD" WHERE "MANDT" = 'A0 AND "VBELN" IN ('A	34.995.778

Figure 141: SQL Statement Analysis Sub-Monitors

These sub-monitors in the DBA Cockpit help you analyze SQL statements.

The following table summarizes the sub-monitors, their data sources, and descriptions.

SQL Statement Analysis Sub-Monitors

Sub-Monitor	Data Source	Remarks
Shared Cursor Cache	shared SQL area	This sub-monitor provides access to SQL requests processed by your database.
Table Access	several views, for example, V\$SQL and V\$SYSSTAT	This monitor displays the shared cursor cache from the viewpoint of the tables accessed. This helps you to identify performance problems for a table rather than for a statement, such as a missing index on a table.
Column Usage	SYS.COL_USAGE\$	This lets you monitor the usage of predicates on columns (equal, like, and so on) in select statements.
System Statistics for CBO	DBMS_STATS package, SYS.AUXSTAT\$	You can collect I/O and CPU statistics for the cost-based optimizer.

Statistical Information

The Statistical Information monitor collects the following sub-monitors:

- SGA Monitor
- PGA Monitor
- Undo Statistics
- Performance Database
- DB Overview Script
- System Metrics
- System Summary Metrics
- File Metrics
- Wait Class Metrics



Figure 142: Statistical Information Sub-Monitors

These sub-monitors in the DBA Cockpit show statistical information.

The following table summarizes the sub-monitors, their data sources, and descriptions.

Statistical Information Sub-Monitors

Sub-Monitor	Data Source	Remarks
PGA Monitor	several views, for example, GV\$PGASTAT	This monitors the Program Global Area.
SGA Monitor	several GV\$\$SGA* and GV\$DB* views	This monitors the System Global Area.
Undo Statistics	GV\$UNDO- STAT	This sub-monitor lets you check the undo statistics.
Performance Database		Gives you an overview of the load and performance of the Oracle instance.
DB Overview Script	SQL script	Gives you statistical information based on an underlying SQL script.
System Metrics	GV\$\$SYSMET- RIC_HISTORY	Gives you metrics on the Oracle database system.

System Summary Metrics	GV\$SYSMETRIC_HISTORY	Gives you system summary metrics on the Oracle database system.
File Metrics	GV\$FILEMETRIC_HISTORY	Gives you file metrics.
Wait Class Metrics	GV\$WAITCLASSMETRIC_HISTORY	Gives you wait class metrics on the Oracle database system.

Feature Monitoring

The Feature Monitoring monitor collects the following sub-monitors:

- Automatic Segment Space Management
- Online Redefinition
- Resumable Space Allocation
- Parallel Query
- Data Guard



Feature Monitoring - Automatic Segment Space Management

Automatic Segment Space Management

DB Name: DEV Started: 24.10.2007

DB Server: TWDF1902 17:47:39

DB Release: 10.2.0.2.0

Tablespaces with ASSM All Tablespaces Tables with ASSM Tables without ASSM

Tablespaces With Automatic Segment Space Management

Name	Block Size	Status	Contents	Extent Management	Allocation Ty...	Segment Space Mn...
SYSAUX	8.192	ONLINE	PERMANENT LOCAL		SYSTEM	AUTO
PSAPSR3	8.192	ONLINE	PERMANENT LOCAL		SYSTEM	AUTO
PSAPSR3700	8.192	ONLINE	PERMANENT LOCAL		SYSTEM	AUTO
PSAPSR3USR	8.192	ONLINE	PERMANENT LOCAL		SYSTEM	AUTO

Figure 143: Feature Monitoring Sub-Monitors

These sub-monitors in the DBA Cockpit show information about Oracle features.

The following table summarizes the sub-monitors, their data sources, and descriptions.

Feature Monitoring Sub-Monitors

Sub-Monitor	Data Source	Remarks
Automatic Segment Space Management		This sub-monitor checks the automatic segment space management (ASSM) of the database. ASSM simplifies and blocks the storage of tables and indexes by replacing linked-list freelists with bitmap freelists, which are faster and more efficient.
Online Redefinition		This monitors the online redefinition of tables in the database. Online redefinition lets you redefine tables – add, rename, or drop columns – while keeping the table fully online and available.
Resumable Space Allocation		Monitoring the resumable space allocation. If a statement is suspended for space allocation reasons, the resumable space allocation feature enables the statement to be resumed, so the work done so far is saved.
Parallel Query	V\$PQ_* views	This sub-monitor checks the occurrence of parallel queries, which are useful in case of full table scans of large tables, creation of large indexes, or bulk inserts, updates, and deletes.
Data Guard	V\$DATABASE	This monitors the Oracle data guard functionality.

Additional Functions

The Additional Functions monitor collects the following sub-monitors:

- SQL Command Editor
- Display GV\$ Views
- Display DBA Tables
- Database Parameters
- Alert Log
- Checkpoints
- Oracle Net

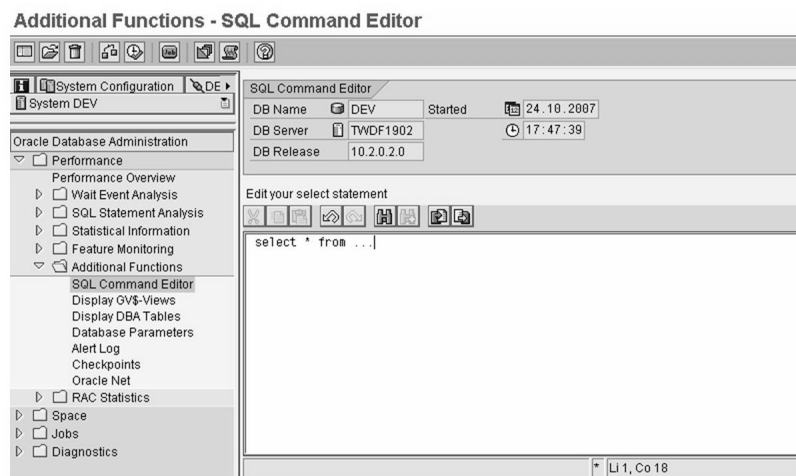


Figure 144: Additional Functions Sub-Monitors

These sub-monitors in the DBA Cockpit show additional functions.

The following table summarizes the sub-monitors, their data sources, and descriptions.

Additional Functions Sub-Monitors

Sub-Monitor	Data Source	Remarks
SQL Command Editor	Oracle views and tables	The sub-monitor consists of an editor screen, where you enter the SQL statement, and a result screen that displays the result of the SQL statement.
Display GV\$ views	Oracle views	Access to the Oracle performance views

Display DBA Tables	DBA tables	Access to the Oracle performance views
Database Parameters	SPFILE, init<DBSID>.ora	Displays the settings in the SPFILE and init<SID>.ora files
Alert Log	alert_<DB-SID>.log	Lets you check the database message log
Checkpoints	alert_<DB-SID>.log	Displays all checkpoints found in the current alert_<SID>.log
Oracle Net		Gives you an overview of the Oracle Net configuration of the Oracle database

RAC Statistics

These sub-monitors show statistics for Oracle Real application clusters (RAC).

Appendix: SAP/Oracle Database Monitor (ST04N)

Prerequisites

In order to fully exploit the functionality of the new SAP/Oracle database monitor of SAP Web Application Server 6.40, a few prerequisites need to be met.

- The new Oracle database monitor was developed for SAP Web Application Server 6.40, but was patched down to SAP Web AS 6.10, according to SAP Note 716000. The Oracle database parameter parallel_min_servers should have a value of at least 10.
- According to SAP Note 706927, the Oracle database administrator should run a few SQL scripts in the database, which will generate necessary snapshot tables, views, and synonyms used by transaction ST04N.
- To have a performance history, ABAP report **RSORAHCL** must be scheduled as periodic batch job via transaction SM36. The scheduling details are stored in table TCOLL. To edit the scheduling data, call either ABAP report **RSCOLL40** or use transaction ST03N.

The Main Monitor

The main monitor, which is the entry screen of SAP/Oracle database monitor ST04N, provides structural and efficiency data on the following items:

- General information
- Data buffer
- Shared pool
- Log buffer
- Calls
- Time statistics
- Redo logging
- Table scans and fetches
- Sorts
- Instance efficiency

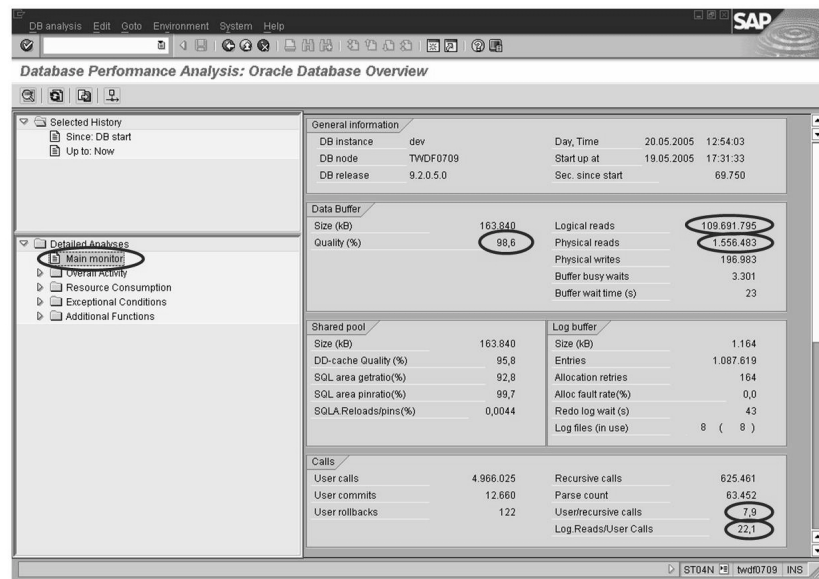


Figure 145: Transaction ST04N: Entry Screen and Main Menu

Many numbers depend on each other. The following checklist names a few key performance figures of your Oracle database.

- The **data buffer quality**, for instance, is based on the ratio of physical reads versus the total number of reads. The lower the ratio the better is the buffer quality. The data buffer quality should be better than 94,0%; the statistics should be based on 15 million total reads. This number of reads ensures that the database is in an equilibrated state.
- Good performance is indicated by a **ratio of user and recursive calls** that is greater than 2. Otherwise, the number of recursive calls compared to the number of user calls is too high. Over time, this ratio always declines because more and more SQL statements are parsed in the meantime.
- If the **number of reads per user call** exceeds 15 blocks per user call, this might indicate an expensive SQL statement.
- Check the value of **time/user call**. Values larger than 15 ms often indicate an optimization issue.
- Compare **busy wait time versus CPU time**. A ratio of 60:40 generally indicates a well-tuned system. Significantly higher values (for example, 80:20) indicate room for improvement.
- The **DD cache quality** should be better than 80%.

In addition to the main menu, transaction ST04N provides specialized sub-monitors for a more detailed analysis.

Monitoring Overall Activity

The Overall Activity monitor collects the following sub-monitors:

- Buffer busy waits
- File system requests
- System / wait events
- Undo statistics
- Automatic segment space management
- Online redefinition tables
- Resumable space allocation
- Parallel query

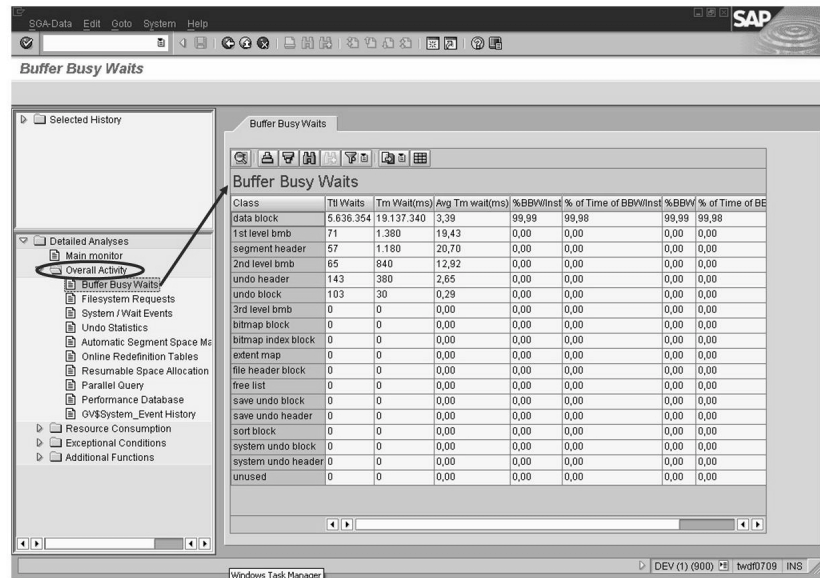


Figure 146: Overall Activity Sub-Monitors

The following table summarizes the sub-monitors, their data sources, and descriptions.

Overall Activity Sub-Monitors

Sub-Monitor	Data Source	Remarks
Buffer busy waits		A buffer busy wait indicates that there are some buffers in the buffer cache that multiple processes are attempting to access concurrently.
File system requests	GV\$FILESTAT	Data file activity has an effect on database performance. This view helps you to identify the frequently used data files and put them on separate disks to avoid contention, if necessary.
System / wait events	GV\$SYSTEM_EVENT	This sub-monitor lets you check wait events and system events in the Oracle database.
Undo statistics	GV\$UNDOSTAT	This sub-monitor lets you check the undo statistics.

Automatic segment space management		This submonitor checks the automatic segment space management (ASSM) of the database. ASSM simplifies and blocks the storage of tables and indexes by replacing linked-list freelists with bitmap freelists, which are faster and more efficient.
Online redefinition tables		This monitors the online redefinition of tables in the database. Online redefinition lets you redefine tables – add, rename, or drop columns – while keeping the table fully online and available.
Resumable space allocation		Monitoring the resumable space allocation. If a statement is suspended for space allocation reasons, the resumable space allocation feature enables the statement to be resumed, so that the work done so far is saved.
Parallel query	V\$PQ_* views	This sub-monitor checks the occurrence of parallel queries, which are useful in case of full table scans of large tables, creation of large indexes, or bulk inserts, updates, and deletes.

Monitoring Resource Consumptions

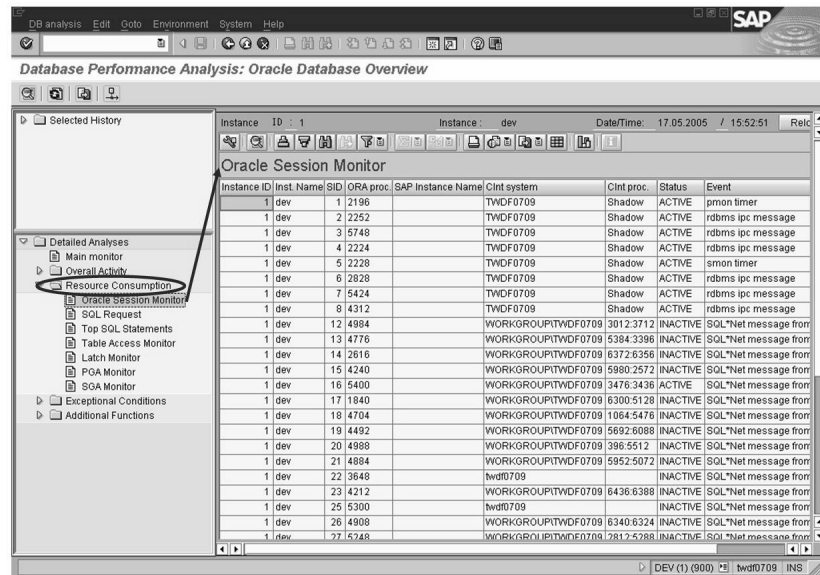


Figure 147: Resource Consumption Sub-Monitors

The following table summarizes the sub-monitors, their data sources, and descriptions.

Sub-monitors Resource Consumption

Sub-Monitor	Data Source	Remarks
Oracle session monitor	several views, for example, V\$Session, V\$Process, and so on	This monitors the Oracle session list and related resource information. In addition, you can see an execution plan and the SQL statement performed by a session.
SQL request	shared SQL area	Access to SQL requests processed by your database
Top SQL statements	SQL script	This sub-monitor lists instance-related information and data history.
Table access monitor	several views, for example, V\$SQL and V\$SYSSTAT	This monitor displays the shared cursor cache from the viewpoint of the tables accessed. This helps you to identify performance problems for a table rather than for a statement, such as a missing index on a table.

Latch monitor	several views, for example, V\$SQL and V\$SYSSTAT	This monitors the latch activity. Processes have to obtain a latch in order to access the data.
PGA monitor	several views, for example, V\$PGASTAT	This monitors the Program Global Area.
SGA monitor	several V\$SGA* and V\$DB* views	This monitors the System Global Area.

Monitoring Exceptional Conditions

Monitoring Single Instance DB

V\$enqueue stat

Content of GV\$ENQUEUE_STAT

Enqueue Type	Enqueue Name	Total Requests	% Requests	Total Waits	% Waits	Total Grants	% Grants
CF	Controlfile Transaction	292,302	40.07	2	0.05	292,299	40.07
TM	DML Enqueue	267,535	36.67	0	0.00	267,538	36.67
TX	Transaction	137,740	18.88	3,366	98.70	137,744	18.88
TT	Temporary Table	8,356	1.14	0	0.00	8,356	1.14
HW		4,696	0.64	8	0.23	4,696	0.64
CU	Bind Enqueue	3,800	0.52	1	0.02	3,800	0.52
FB		3,014	0.41	0	0.00	3,014	0.41
MR	Media Recovery	2,645	0.36	0	0.00	2,645	0.36
TA	Transaction Recovery	2,430	0.33	0	0.00	2,430	0.33
CI	Cross-instance Call Invocation	2,424	0.33	0	0.00	2,424	0.33
TD		2,221	0.30	0	0.00	2,221	0.30
PE		631	0.08	0	0.00	631	0.08
DR	Distributed Recovery	374	0.05	0	0.00	374	0.05
US	Undo Segment, Serialization	372	0.05	0	0.00	372	0.05
TS	Temporary Segment	298	0.04	0	0.00	298	0.04
IS	Instance State	196	0.02	0	0.00	196	0.02
TC		150	0.02	30	0.87	150	0.02
DL	Direct Loader Index Creation	102	0.01	0	0.00	102	0.01
MD		72	0.00	0	0.00	72	0.00
WL	Being Written Redo Log	45	0.00	0	0.00	45	0.00

Figure 148: Exceptional Conditions Sub-Monitors

The following table summarizes the sub-monitors, their data sources and descriptions.

Exceptional Conditions Sub-Monitors

Sub-Monitor	Data Source	Remarks
Enqueue monitor	V\$EN- QUEUE_STAT	This monitors enqueues.
Lock monitor	V\$LOCK* views	This monitors currently active locks.
Database message log	log and trace files	To use this submonitor, make sure that BRTOOLS is installed on the database instance and that all administrative files are maintained correctly. See SAP Notes 80689 and 446172.

Monitoring Additional Functions

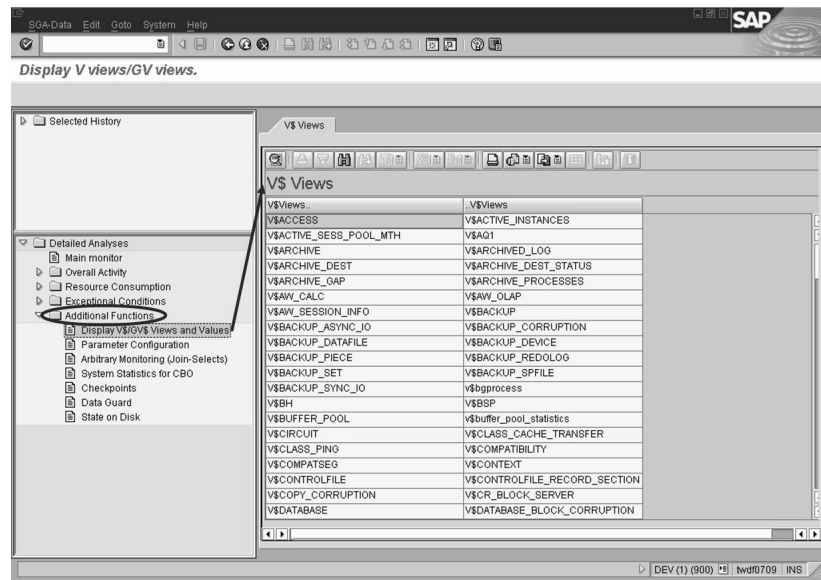


Figure 149: Additional Functions Sub-Monitors

The following table summarizes the sub-monitors, their data sources, and descriptions.

Additional Functions Sub-Monitors

Sub-Monitor	Data Source	Remarks
Display V\$/GV\$ views and values	Oracle views	Access to the Oracle performance views
Parameter configuration	SPFILE, init<DBSID>.ora	Displays the settings in the SPFILE and init<SID>.ora files
Arbitrary monitoring	Oracle views and tables	Views and tables must be of owners SYS or Public.
System statistics for CBO	PL/SQL scripts	The scripts are part of the DBMS_STATS package.
Checkpoints	alert_<DB-SID>.log	Oracle parameter Log_checkpoints_to_alert must be set to TRUE.
Data guard	V\$DATABASE	This monitors the Oracle data guard functionality.
State on disk	DB02	This calls transaction DB02.

Enhanced History Functions

Some sub-monitors provide history data, called snapshots. You even can select the time frame that should be displayed. By default, the time span is from DB start up to now. The following table summarizes all possible options.

Selected History Options

Since	Up to	Remark
DB start	now	Displays the changes from database start to the current time
DB start	snapshot	Displays the changes from database start to the selected snapshot time
snapshot	now	Displays the changes from your selected snapshot to the current time
snapshot1	snapshot2	Displays the changes between your selected snapshots

To choose another time frame, double-click the corresponding delimiter in the *Selected History*.

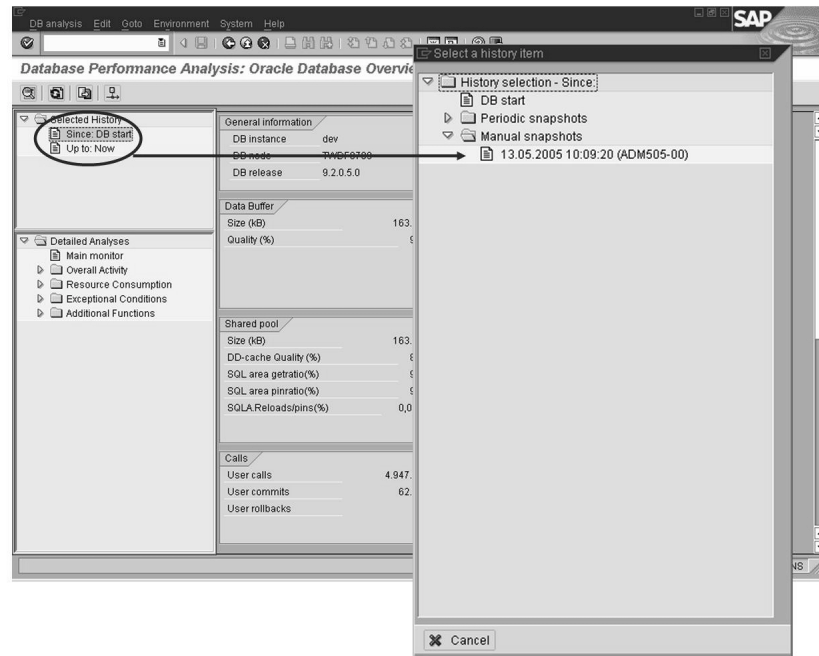


Figure 150: Selecting the History Time Frame for Enhanced History Function

Enhanced History Function: Additional Information

As mentioned earlier, some sub-monitors provide history data. The history is stored in corresponding SAP tables. Oracle performance views, which are displayed in lowercase letters, have such corresponding tables. The tables are updated by ABAP report **RSORAHCL** (or RSORAHIST). History data of global views is stored in SAP tables GVD_*. Note that the Oracle history data is not stored in SAP table MONI, which contains the history of SAP performance data.

Exercise 16: Navigation in the DBA Cockpit Performance Monitor

Exercise Objectives

After completing this exercise, you will be able to:

- Navigate in the DBA Cockpit database performance monitor

Business Example

Using transaction DBACOCKPIT, you want to look at the current Oracle sessions.

Task 1:

Prepare a list of all current Oracle sessions in the state ACTIVE using the Oracle session monitor.

1. Call the DBA Cockpit performance monitor.
2. Call the Oracle session monitor.
3. Filter the listed data so that only active sessions are displayed.

Task 2:

Prepare a list of all current Oracle sessions using the Oracle performance view v\$session.

1. Call the DBA Cockpit performance monitor.
2. Call the monitor entry *Display GV\$ views*.
3. Scroll down and open the view v\$session.

Solution 16: Navigation in the DBA Cockpit Performance Monitor

Task 1:

Prepare a list of all current Oracle sessions in the state **ACTIVE** using the Oracle session monitor.

1. Call the DBA Cockpit performance monitor.
 - a) Call transaction DBACOCKPIT or ST04.
2. Call the Oracle session monitor.
 - a) In the navigation bar on the left-hand side, choose *Performance* → *Wait Event Analysis* → *Session Monitor*. Double-click the monitor and a list of Oracle sessions will be displayed in the content area on the right hand side.
3. Filter the listed data so that only active sessions are displayed.
 - a) Select the header line of the *Status* column.
 - b) Choose the *Set filter* button from the function tool bar.
 - c) In the dialog box, enter **ACTIVE** in the *Status* field.
 - d) Choose *Execute* and you will only see active sessions.

Task 2:

Prepare a list of all current Oracle sessions using the Oracle performance view v\$session.

1. Call the DBA Cockpit performance monitor.
 - a) Call transaction DBACOCKPIT or ST04.
2. Call the monitor entry *Display GV\$ views*.
 - a) Chose *Performance* → *Additional Functions* → *Display GV\$ views* and double-click the monitor entry.
3. Scroll down and open the view v\$session.
 - a) Optionally, you can filter data. For instance, choose the *Status of the session* column. Open the context menu (right-click) and choose *Set Filter*.



Lesson Summary

You should now be able to:

- Use the DBA Cockpit to analyze the performance of the Oracle database
- Describe how the DBA Cockpit supports the database features of Oracle 9i and Oracle 10g
- Use the enhanced history functionality of the DBA Cockpit



Unit Summary

You should now be able to:

- Use the DBA Cockpit to analyze the performance of the Oracle database
- Describe how the DBA Cockpit supports the database features of Oracle 9i and Oracle 10g
- Use the enhanced history functionality of the DBA Cockpit



Test Your Knowledge

1. Which of the following statements is correct?

Choose the correct answer(s).

- ☐ A The new database monitor makes database backups obsolete.
- ☐ B Transaction ST04N provides access to Oracle performance views V\$ and DBA.
- ☐ C The new database monitor automatically optimizes SQL statements.
- ☐ D ST04N only supports Oracle databases.



Answers

1. Which of the following statements is correct?

Answer: B, D

In addition to V\$ and DBA views and tables, GV\$ views can be monitored. Indeed, the new Oracle database monitor is only available for Oracle-based SAP systems.

Unit 7

Appendix: Monitoring of the database instance

Unit Overview

This unit introduces how the Oracle wait interface could be used to detect performance problems and the Automatic Workload repository a new feature of Oracle 10g..



Unit Objectives

After completing this unit, you will be able to:

- Use Oracle wait event to determine what a session is currently doing or what it is waiting for
- Draw conclusions about the potential for optimization by using the wait events
- Trace the wait events of an Oracle session and determine the most important wait events in previous periods
- Explain the potential of the Automatic Workload Repository to monitor database performance
- Use the DBA Cockpit to analyze the Automatic Workload Repository

Unit Contents

Lesson: Oracle Wait Interface.....	436
Exercise 17: Monitoring Oracle Wait Events.....	459
Lesson: Automatic Workload Repository and Histories.....	462

Lesson: Oracle Wait Interface

Lesson Overview

This lesson introduces the Oracle Wait Interface, which can be used to draw conclusions about performance bottlenecks. We will discuss the most common busy wait situations and the potential for optimization.



Lesson Objectives

After completing this lesson, you will be able to:

- Use Oracle wait event to determine what a session is currently doing or what it is waiting for
- Draw conclusions about the potential for optimization by using the wait events
- Trace the wait events of an Oracle session and determine the most important wait events in previous periods

Business Example

Your users complain about low performance of your system. To determine where the low performance may originate, you want to analyze the Oracle wait events. You can then determine whether the performance problems are caused by database locks, disk accesses, latches, or another cause.

Introduction to the Wait Interface

Many processes work together in an Oracle instance. During operation of the instance, the shadow processes often need to stop and wait for various reasons, such as:



- They have no work to do.
- A resource is not available.
- They need to wait for another shadow process to perform a prerequisite task.

As an example, let's look at the cooperation between the shadow process and the Oracle Log Writer (LGWR). During a shadow process copies redo information into the redo log buffer, the LGWR may be waiting for work. When the user commits, the LGWR must write the redo information and the commit marker to the online redo log file, while the shadow process has to wait. When the log file I/O is complete, the LGWR sends a signal to the shadow process that it can now begin its next transaction, because the commit operation has completed. The Log Writer then waits again for work.

Oracle defined a list of every possible event that an Oracle process could be waiting for. At any moment, every Oracle process that is not busy is waiting for one of these events to occur. These **wait events** are characteristic parts of the Oracle kernel source code that can contain Oracle sessions during execution.

Suppose a dedicated server process is waiting for an application to submit an SQL statement for execution. This wait event is called **SQL*Net message from client**. Another dedicated server process might be waiting for a row-level lock on a table to be freed so that an UPDATE statement can continue. That wait event is called **enqueue**.

The database instance collects statistics that provide information about what events processes are waiting on and summary information of how much time each Oracle process has spent waiting on each type of wait event for the duration of the process. In addition, an Oracle process could write detailed wait event data to a trace file.

Wait Events

By determining the wait event, you can determine what a session is currently doing or what it is waiting for. If a report takes four hours to complete, for example, the wait event interface will tell you how much of that four hours was spent waiting for disk reads caused by full table scans, disk reads caused by index lookups, latch contention, and so on.

The wait event interface gives you much more information to work with than cache hit ratios do. The wait event interface gives both breadth and depth in the information it provides you. You get data that can touch upon many different areas of your database, such as disk I/O, latches, parallel processing, network traffic, checkpoints, and row-level locking. At the same time, you get incredible detail, such as the file number and block number of a block being read from disk, or the name of a latch being waited on, along with the number of retries.

The wait event interface will not always give you all of the information you need to diagnose and solve a problem, but it will certainly point you in the right direction. You might think the buffer cache is too small because the cache hit ratio is only 70%, but in fact the application's slow response time could be caused by latch contention in the shared pool, a bottleneck in the log writer, or a number of other things.

Wait Events and Database Performance

Detailed analysis of the wait events is very useful in case of performance problems. The database response time is largely determined by :



- Waiting within the framework of wait events
- CPU consumption

When an Oracle process is waiting for a signal to start, the operating system will not schedule the process to run on a CPU. In operating-system terms, the process is blocked, and not able to run. When the process receives the signal to start, the operating system changes the status of the process from blocked to runnable, and the process will be scheduled to run as soon as possible. The process has to wait at least until the operating system scheduler runs next, and possibly longer if there are higher-priority processes to run. The delay from when a process is posted until it begins to run contributes to the database response time. Minimizing this delay is an important part of performance tuning.

In well-tuned systems, the wait events make up about 60% of the response time. Otherwise, the proportion may be much higher, which has a negative effect on the response times. Wait event tuning can therefore frequently bring about a significant improvement in database performance.



Note: For more information on the distribution between wait event time and CPU time, refer to the *Performance Overview* screen of the DBA Cockpit, transaction DBACOCKPIT (Busy wait time and CPU time session).

Looking at all the wait events that have accumulated since the database was started enables you to draw conclusions as to the extent of the potential for optimization and where you need to focus to increase global database performance. If sporadic performance bottlenecks occur, analyzing the wait events that occurred during the problematic period can also be very helpful.

By determining the wait event, you can determine what a session is currently doing or what it is waiting for. If there are performance problems, you can determine whether they are caused by database locks, disk accesses, latches, or another cause.

Limitations of the Wait Interface

If an Oracle process has work to do but must wait for something to happen before it can continue, the process is waiting on a non-idle wait event. If a process has nothing to do, it is waiting on an idle wait event. When a process is busy, there will be no information in the wait event interface (since the process is not waiting).

The wait event information provided by the Oracle instance contains detailed information about how many times the process had to wait and how much time was spent waiting for specific events to occur. But no information is available about the time periods in which the process requested use of the CPU. This means that the Wait Interface is not able to provide information about:



- Time spent using the CPU
- Time spent waiting for a CPU to become available
- Time spent waiting for requested memory to be swapped back in to physical memory

This is important to keep in mind. You could be troubleshooting a very slow SELECT statement and learn from the wait event interface that the process does not have significant wait events. This could lead you to think that the statement is as optimal as it can be, and that it just takes a long time to run. In fact, however, the query might be performing huge numbers of logical reads and the number of buffer gets could be reduced by rewriting the query.

When Oracle needs to access a data block and the block is already in the buffer cache, a logical read occurs with no physical read. The process is able to read the block without the occurrence of any wait events. Large amounts of CPU time could be consumed on significant numbers of logical reads, and the wait event interface will have nothing to show for the elapsed time.

Common Wait Events

Although there are many different types of wait events, the majority of them come up very infrequently or tend not to be significant. In practice, only a few dozen wait events tend to be of interest to most database administrators. The rest are rather obscure, or pertain to Oracle features not in use, or occur so infrequently that you do not need to worry about them. You will see different wait events surfacing in different environments, based on which Oracle features have been implemented.

As of Oracle 10g, several improvements have been implemented to help analyze and understand wait situations. To improve the overview of wait situations, Oracle 10g groups wait events into different wait classes:

- Other
- Application
- Configuration
- Administrative
- Concurrency
- Commit

Idle

Network

User I/O

System I/O

There is a difference between cases when the Oracle process is waiting because there is currently nothing for it to do (idle waits), and when the Oracle process wants to run but first has to wait for a resource that is not yet available (busy waits). **Total wait time** describes the sum of idle wait time and busy wait time.



The Most Common Busy Wait Situations

Wait Event	Meaning
db file sequential read	Waiting for parallel blocks to be read from the hard disk
db file parallel read	Waiting for blocks to be read in parallel from the disk
db file scattered read	Waiting for several blocks to be read by the disk
log file sync	Waiting until the LGWR has written all data from the redo buffer into the online redolog during a COMMIT or ROLLBACK
log buffer space	Waiting for free space in the redo buffer
log file switch	Waiting for a redo log switch
log file parallel write	LGWR waiting for blocks to be written to disk
buffer busy waits / read by other session	Waiting for a block because it is currently being imported or changed by another session
write complete waits	Waiting until the DBWR has written a necessary block to the disk
enqueue / enq:<type>-<desc.>	waiting for an Oracle lock
library cache pin / library cache lock	Waiting for exclusive access to data of the library cache (shared SQL area)
row cache lock	Waiting for exclusive row cache access
db file parallel write	DBWR is waiting until blocks have been written to disk

Wait Event	Meaning
direct path read	Waiting when reading buffers from disk into the PGA
direct path write	Waiting upon writing buffers directly from PGA to disk
free buffer waits	Waiting when searching for free buffer blocks

Idle wait situations include:



- **SQL*Net message from client** (the process is waiting for an SQL statement from the client, for example, the SAP work process)
- **rdbms ipc message** (the process is waiting for a statement from the RDBMS)
- **dispatcher timer, virtual circuit status, pmon timer, smon timer, WMON goes to sleep, Null event**

Idle events, as defined by Oracle, are wait events that are reported when the Oracle process has nothing to do.

In addition, it is useful to distinguish between idle events as defined by Oracle and idle events as defined by SAP. Idle events, as defined by SAP, are the events that do not have any influence over database response time. These include, for example, **db file parallel write**, because the related Database Writer (DBWR) processes occur asynchronously and a client inquiry does not have to wait for this. Those events whose times are completely covered in the context of other non-idle events (for example, **db file parallel write**, which is contained in the **log file sync**) are also considered idle events from an SAP point of view.

All events that are not defined by SAP as idle events are entered in the database response time and are included in the core of a database performance analysis. SAP defines the following idle wait situations:



- db file parallel write
- Log archive I/O
- log file parallel write
- log file sequential read

In general, all wait events for background processes are defined by SAP as idle events. The non-idle waits that are allotted to background processes (such as db file sequential read) are therefore not relevant from an SAP perspective.

Collecting Wait Event Information

The Oracle Wait Interface consists of four dynamic performance views (V\$-views).

These V\$-Views can be used to access wait event information. The following is important in this context:

V\$SYSTEM_EVENT

Wait events accumulated system-wide since the database was started

V\$SESSION_EVENT

Wait events accumulated per session since the database was started

V\$SESSION_WAIT

Current wait events

V\$EVENT_NAME

Name and parameter of the wait events

In addition to direct access at Oracle level, you can also use the DBA Cockpit (transaction DBACOCKPIT) and choose *Performance* → *Additional Functions* → *Display GV\$ views* to access these views. Sometimes you have to use direct Oracle queries because you can not use the SAP system if extreme performance problems occur.

Monitoring Wait Events

The database response time consists of the wait time and the CPU time. In well-tuned systems, the wait events make up about 60% of the response time. Otherwise, the proportion may be much higher, which has a negative effect on the response times. Wait event tuning can therefore frequently bring about a significant improvement in database performance.

The distribution between wait event time and CPU time could be monitored using the DBA Cockpit (transaction DBACOCKPIT). The Busy wait time and the CPU time session are displayed on the *Performance Overview* screen of transaction DBACOCKPIT. The approach to determine the possible tuning potential is displayed in the following figure.

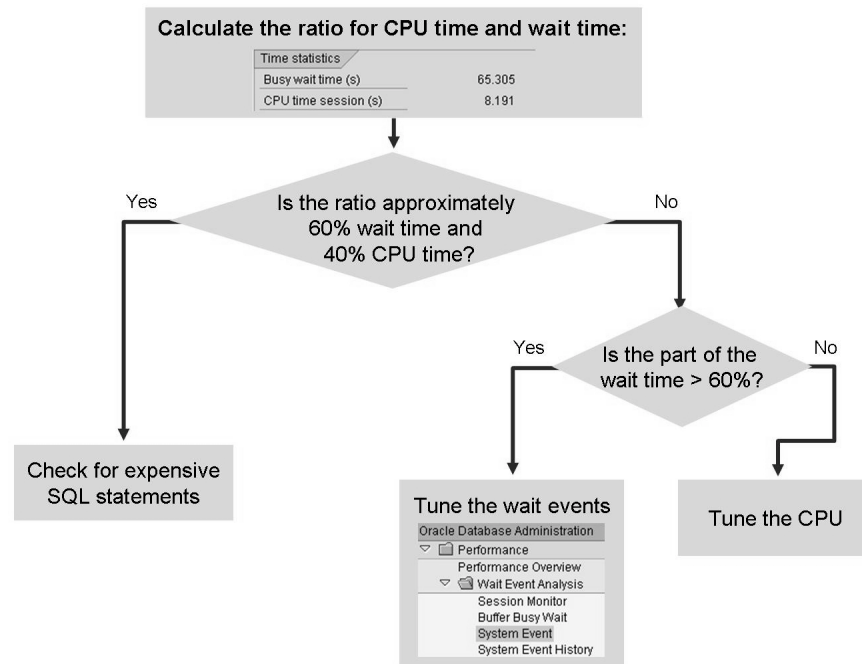


Figure 151: Oracle Tuning Road Map

The most important wait event in previous periods could be determined using the System Events sub-monitor in the DBA Cockpit

This sub-monitor lets you check the following wait events and system events in the Oracle database:



- Busy waits summary
- Wait event details
- Oracle view GV\$SYSTEM_EVENT

The Busy Waits Summary monitor displays a summary of busy waits depending on the database processes. The session type BACKGROUND represents an Oracle process, and the session type USER represents processes from application sessions. For each process type, the totals for the busy wait time and total wait time are calculated.



Busy Waits Summary						
Wait Event Details						
GV\$SYSTEM_EVENT						
Busy Wait Summary						
Session type	Username	PName	Sessions	Busy wait time (ms)	Total wait time (ms)	Busy W %
BACKGROUND		CKPT	1	316.135	756.954.918	0,04
BACKGROUND		DBW0	1	664.049	756.192.225	0,09
BACKGROUND		DBW1	1	58.302	756.853.780	0,01
BACKGROUND		LGWR	1	551.409	756.747.036	0,07
BACKGROUND		MMAN	1	0	757.009.587	0,00
BACKGROUND		MMNL	1	8	757.070.094	0,00
BACKGROUND		MMON	1	11.051	756.994.771	0,00
BACKGROUND		PMON	1	31	756.996.759	0,00
BACKGROUND		PSP0	1	0	757.012.378	0,00
BACKGROUND		QMNC	1	754.877.650	754.877.650	100,00
BACKGROUND		RECO	1	257	596.783.921	0,00
BACKGROUND		SMON	1	727.338	731.986.574	0,10
USER	SAPSR3	User	30	9.590.979	1.596.466.789	0,60
TOTAL			42	766.797.209	10.491.946.482	7,31

Figure 152: Busy Waits Summary Monitor

The Wait Event Details monitor provides details about each wait event. This view does not include wait events from background process or idle processes. Only wait events from non-idle processes are listed. Idle events are defined in table `ORA_IDLE_EVENTS`.

CPU used by this session is added to the list as a special event.

The list is sorted by wait time in descending order. Additional key figures are:

- Percentage of the event in relation to all non-idle wait times (without consideration of **CPU used by this session**)
- Percentage of the event in relation to total wait times (including **CPU used by this session**)
- Number of waits
- Number of time-outs
- Average wait time (per wait)



GV\$SYSTEM_EVENT						
Wait Event Details						
Event	Wait time (ms)	% of non-idle	% of tot. resp.	Waits	Timeouts	Avg.WT (ms)
Total	45.886.291	100,00	100,00	29.126.667	80.504	2
db file scattered read	12.514.984	31,14	27,27	3.136.273	0	4
db file sequential read	8.157.963	20,30	17,78	4.266.365	0	2
enq: TX - row lock contention	7.616.816	18,95	16,60	32.534	678	234
SQL*Net more data to client	7.411.008	18,44	16,15	183.739	0	40
CPU used by this session	5.699.460	0,00	12,42	0	0	0
log file sync	3.393.413	8,44	7,40	189.512	8	18
read by other session	760.613	1,89	1,66	261.336	0	3
cursor: pin S wait on X	59.418	0,15	0,13	5.658	4.430	11
enq: HW - contention	54.460	0,14	0,12	3.219	0	17
control file sequential read	48.955	0,12	0,11	269.636	0	0
SQL*Net message to client	43.083	0,11	0,09	20.525.713	0	0
log file switch completion	39.638	0,10	0,09	1.156	0	34
enq: TX - index contention	13.509	0,03	0,03	1.172	0	12
latch free	11.876	0,03	0,03	944	0	13
SQL*Net break/reset to client	10.871	0,03	0,02	127.661	0	0
latch: cache buffers chains	9.203	0,02	0,02	9.775	0	1
Data file init write	9.020	0,02	0,02	2.586	0	3
buffer busy waits	7.868	0,02	0,02	15.755	0	0
library cache load lock	4.623	0,01	0,01	682	0	7

Figure 153: Wait Event Details Monitor

The GV\$SYSTEM_EVENT monitor provides the raw information of the Oracle view GV\$SYSTEM_EVENT. The list shows events and wait times per instance ordered by the event's total wait time (descending). If the current system is in a RAC environment, you will see the breakdown of event wait times into different instances. Additional key figures are:

- Percentage of the instance's event wait time in relation to all instance's event wait times
- Number of waits
- Number of timeouts
- Average wait time (per wait)



Busy Waits Summary					
Wait Event Details					
GV\$SYSTEM_EVENT					
GV\$SYSTEM_EVENT					
Event	Wait time (ms)	Wait% InstEvt.	Waits	Timeouts	Avg.WT (ms)
rdbms ipc message	7 019 917 966	100,00	3 019 863	2 594 160	2 325
rdbms ipc message	7 019 917 966	100,00	3 019 863	2 594 160	2 325
SQL*Net message from client	3 590 493 613	100,00	20 525 658	0	175
SQL*Net message from client	3 590 493 613	100,00	20 525 658	0	175
pmon timer	756 996 728	100,00	260 407	260 397	2 907
pmon timer	756 996 728	100,00	260 407	260 397	2 907
Streams AQ: qmn coordinator idle wait	754 877 637	100,00	55 826	28 930	13 522
Streams AQ: qmn coordinator idle wait	754 877 637	100,00	55 826	28 930	13 522
Streams AQ: qmn slave idle wait	754 828 992	100,00	26 897	0	28 064
Streams AQ: qmn slave idle wait	754 828 992	100,00	26 897	0	28 064
smon timer	731 259 236	100,00	2 710	2 469	269 837
smon timer	731 259 236	100,00	2 710	2 469	269 837
Streams AQ: waiting for time management or cleanup tasks	491 795 391	100,00	176	176	2 794 292
Streams AQ: waiting for time management or cleanup tasks	491 795 391	100,00	176	176	2 794 292
db file scattered read	13 114 163	100,00	3 284 852	0	4
db file scattered read	13 114 163	100,00	3 284 852	0	4
db file sequential read	8 283 961	100,00	4 315 450	0	2
db file sequential read	8 283 961	100,00	4 315 450	0	2
enq: TX - row lock contention	7 616 816	100,00	32 534	678	234
enq: TX - row lock contention	7 616 816	100,00	32 534	678	234

Figure 154: GV\$SYSTEM_EVENT Monitor

As of Oracle 10g, you can use V\$ACTIVE_SESSION_HISTORY to display historical data of active sessions. Each second, the system checks whether a session is currently active (that is, it is waiting for a non-idle wait event or is consuming CPU). If it is active, an entry is created in V\$ACTIVE_SESSION_HISTORY.

Optimization of Individual Wait Events

To find wait events with high potential for optimization, break down the wait time into the five non-idle wait events with the most impact on performance.

The following list provides explanations, rules of thumb, analysis steps, and optimization options for the most important wait events. In addition, the three relevant wait event parameters are specified under *Parameter*.



Note: Implementing these recommendations requires detailed knowledge of Oracle. In addition, many of the wait events below can also be optimized with SQL statement tuning (see SAP Note 766349).

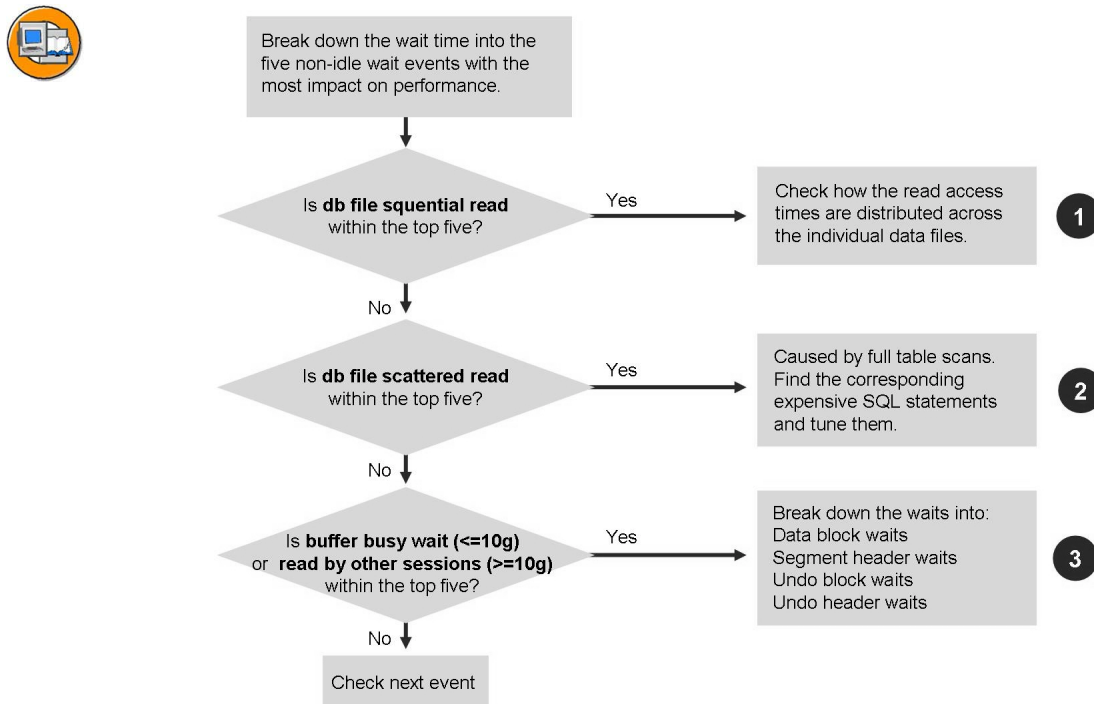


Figure 155: Tuning Wait Events

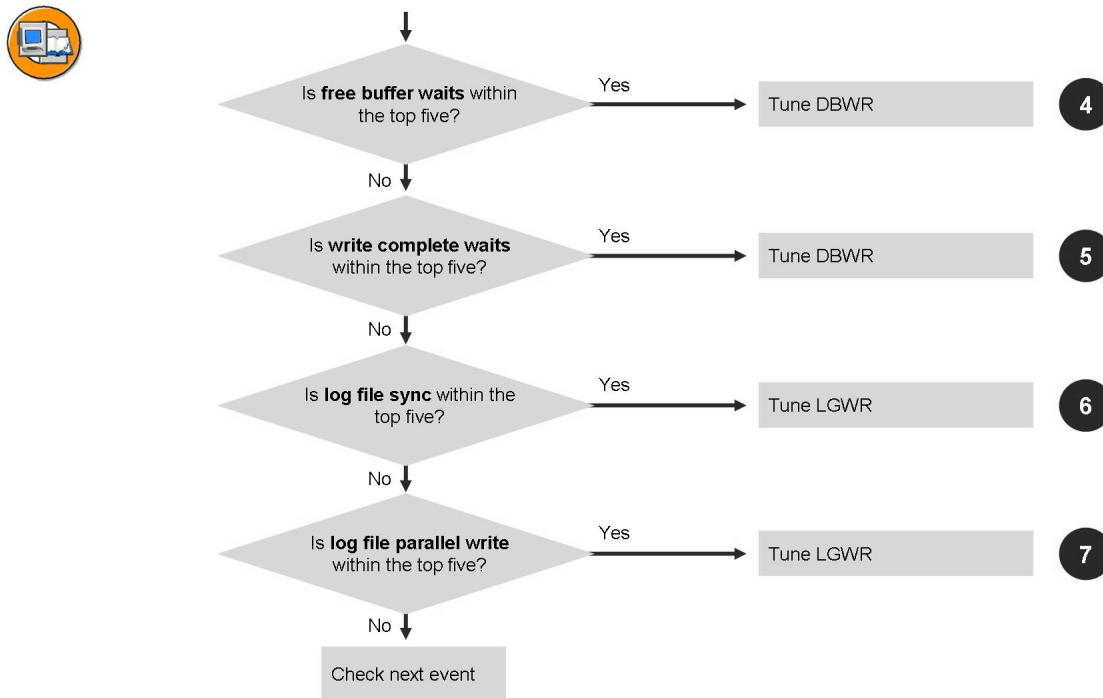


Figure 156: Tuning Wait Events (2)

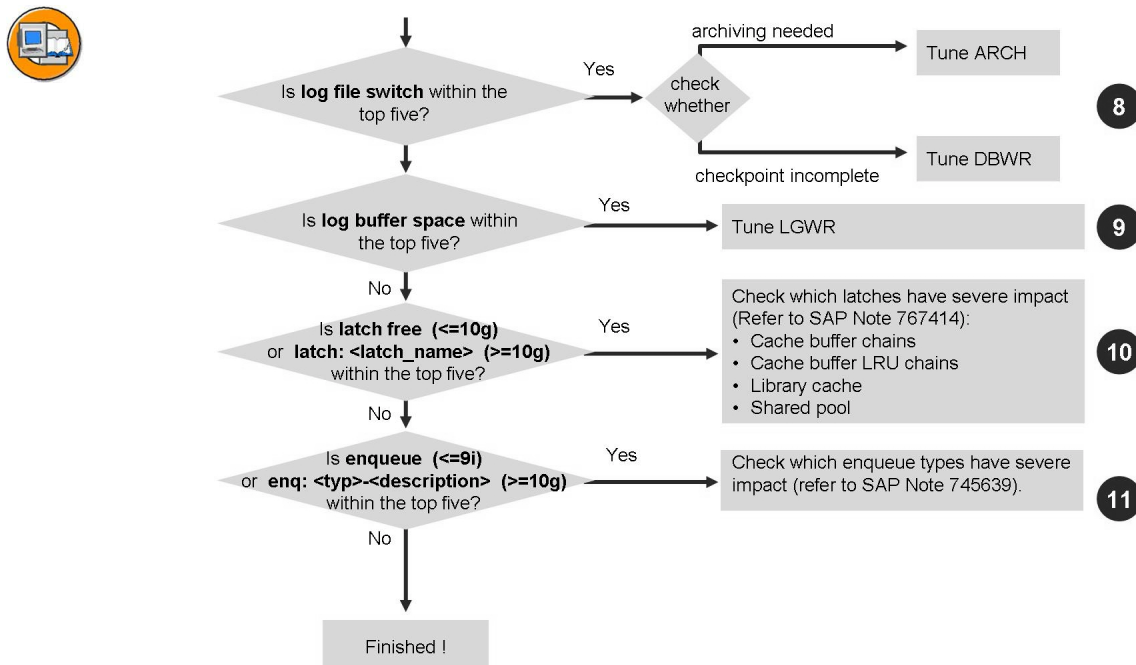


Figure 157: Tuning Wait Events (3)**1. db file sequential read**

- Meaning: Waiting for parallel blocks to be read from the hard disk
- Parameters: File number/block number/1
- Rule-of-thumb: The average value for the wait time in V\$SYSTEM_EVENT should not exceed 20 ms (or 15 ms for read-write-cache- memory).
- Optimization steps:

Using V\$FILESTAT or the DBA Cockpit, choose *Performance* → *Wait Event Analysis* → *Filesystem requests* to check how the read access times are distributed across the individual data files. If individual data files have a poor access time due to very high access rates, you can solve the problem by improving the distribution of the critical objects (for example, by using reorganization or even distribution of the data files on all disks).

In cooperation with your hardware partner, you should also check whether the hardware works correctly and the setup is optimum. Also refer to SAP Note 793113 for an optimal I/O configuration.



Note: Even apparently good access times (8 ms, for example) may result in poor performance if the times previously observed were significantly lower (2 ms, for example). You should therefore always compare the current average values with past values (for example, using report /SDF/RSORAVSE).

In addition to the average access time, the number of **db file sequential read** requests are also significant. Therefore, use transaction DBACOCKPIT to check whether the buffer pool hit rate is not yet sufficiently high after an extensive SQL statement tuning (SAP Note 766349) and increase the Oracle DB_CACHE_SIZE parameter if necessary, taking into account the address space restriction in the operating system and the available main memory (see also SAP Note 618868).

2. db file scattered read

- Meaning: Waiting for several blocks to be read by the disk
- Parameters: File number/block numbers/requests
- Optimization steps:

The **db file scattered read** wait event is caused by full table scans or (more rarely) by index fast full scans. These should be avoided if possible in the SAP environment. Therefore, check which statements execute full table

scans and optimize these. There is no special overview that executes full table scan statements. A high number of disk reads in the SQL cache (DBA Cockpit: *Performance* → *SQL Statement Analysis* → *Shared Cursor Cache*) is normally a good indication. You should therefore specifically check all SQL statements that are at the very top in the SQL cache in relation to disk reads for full table scans.

In addition, as of Oracle 9i, the V\$SQL_PLAN view is available, which allows you to determine SQL statements that use full table scans or index fast full scans:

```
SELECT SQL_TEXT FROM V$SQLAREA SA WHERE
(SA.ADDRESS, SA.HASH_VALUE) IN
(SELECT ADDRESS, HASH_VALUE FROM V$SQL_PLAN
WHERE OPERATION = 'TABLE ACCESS' AND
      OPTIONS = 'FULL' OR
      OPERATION = 'INDEX' AND
      OPTIONS LIKE 'FAST FULL%') ;
```

As of Oracle 10g, V\$SEGMENT_STATISTICS contains information about the number of full table scans or Index fast full scans per segment:

```
SELECT * FROM
( SELECT OWNER, OBJECT_NAME, VALUE
  FROM V$SEGMENT_STATISTICS
 WHERE STATISTIC_NAME = 'segment scans'
 ORDER BY VALUE DESC )
WHERE ROWNUM <=20 ;
```

3. **buffer busy waits** (Oracle <= 9i)

read by other session / buffer busy waits (Oracle >= 10g)

- Meaning: Waiting for a block because it is currently being imported or changed by another session
- Parameters: File number/block number/ID
- Rule-of-thumb: The average value for the wait time in V\$SYSTEM_EVENT should not exceed 40 ms (or 15 ms for read-write-cache memory).
- Optimization steps:

If the rule of thumb is exceeded, this is generally due to problems in the I/O area. You should therefore also carry out the same checks in this case as described in the section **db file sequential read**.

The ID specified as a third parameter gives more exact information as to why the block cannot be accessed. It is generally the case that if the first number is a 1, just the block is read. If the first number is a 2, the block is kept in an incompatible mode.

As of Oracle 10g, the name of the wait event tells you whether it is a read wait event (read by other session) or a compatibility wait event (buffer busy waits).

Oracle view V\$WAITSTAT contains an overview of how often buffer busy waits occurred in the individual block types since the database was started, as well as the average length of the waits. Depending on the type of the block, take the following measures.

Data block:

If INSERTs run onto the buffer busy waits, the number of the FREELISTS can be increased for the affected segment. This change can be carried out dynamically - you do not need to reorganize the object. Otherwise you must optimize the SQL statement, the application and/or the I/O subsystem. You may need to increase the DB_CACHE_SIZE Oracle parameter.

Segment header:

Use several FREELIST GROUPs for the segment concerned. You can only change the number of FREELIST GROUPs when reconstructing the object.

Undo block:

The I/O-times for the rollback segments must be optimized.

Undo header:

There are too few rollback segments. Increase the number of the rollback segments to at least a quarter of the number of work processes.

Buffer busy waits that have a significantly high average wait time may be the consequence of a simultaneous **archiver stuck**. In this case, the buffer busy waits are only a consequence of the problem, and do not require further analysis.

4. free buffer waits

- Meaning: Waiting for the DBWR to write dirty blocks to disk so that these can be replaced by new blocks
- Parameters: File number/ block number/ID
- Rule-of-thumb: Should not be in the top 10 of V\$SYSTEM_EVENT
- Optimization steps:

This indicates that the DBWR processes are not writing changed blocks to the disk quickly enough. Therefore, check the Oracle I/O configuration as described in SAP Note 793113. Also, ensure that the Oracle buffer pool (DB_CACHE_SIZE) is large enough (refer to SAP Note 789011).

If you determine that not all DBWR processes are continually active, despite very large free buffer waits, you can reduce the Oracle parameter FAST_START_MTTR_TARGET as a test, to force the DBWR processes to become more active. However, bear in mind that reducing this parameter causes a greater write load on the DBWR processes, which can be counterproductive.

5. **write complete waits**

- Meaning: Waiting until the DBWR has written a necessary block to the disk
- Parameters: File number/block number/ID
- Rule-of-thumb: Should not be in the top 10 of V\$SYSTEM_EVENT
- Optimization steps:

You must tune the DBWR performance. Check whether the I/O subsystem can be tuned, whether the database buffer pool (DB_CACHE_SIZE) is set too small, and whether it would be useful to define several DBWR processes.

6. **log file sync**

- Meaning: Waiting until the LGWR has written all data from the redo buffer into the online redolog during a COMMIT or ROLLBACK
- Parameters: Buffer number/-/-
- Rule-of-thumb: The average value for the wait time in V\$SYSTEM_EVENT should not exceed 15 ms.
- Optimization steps:

You must tune the LGWR performance. Check that the I/O subsystem is working correctly and that the settings are optimized. Note that no other files with access rights should be saved on the disks with the online redo logs, and that high-speed disks and hardware-stripping should be used for the disks with the online redo logs. Due to adverse effects on performance, a standard RAID 5 should not be used for the online redo log disks.



Caution: As of Oracle 10g, Oracle gives you the option of controlling the performance of **log file sync** using the parameter COMMIT_WRITE. Because the consistency of the application

can no longer be guaranteed if settings are changed, changing this parameter in a way that deviates from the standard system is generally not allowed in the SAP environment.

7. **log file parallel write**

- Meaning: LGWR waiting for blocks to be written to disk
- Parameters: File/block/I/O requests
- Optimization steps:

Tune the LGWR I/O as described in the context of the **log file sync** wait event.

8. **log file switch**

- Meaning: Waiting for a redo log switch
- Optimization steps:

In the case of **log file switch (archiving needed)**, check whether an archiver stuck has occurred (see SAP Note 391)

In the case of **log file switch (checkpoint incomplete)**, tune the DBWR to solve the checkpoint not complete situation (see SAP Note 79341).

9. **log buffer space**

- Meaning: Waiting for freespace in the redo buffer
- Optimization steps:

If the size of the redo buffer (parameter LOG_BUFFER) is less than 1 MB, you can increase this memory area to 1 MB. However, frequent log buffer space wait events are generally triggered by I/O problems with the LGWR. You should therefore refer to the tuning notes for the **log file sync** wait event.

10. **latch free** (Oracle <= 9i)

latch: <latch_name> / **latch free** (Oracle >= 10g)

- Meaning: Waiting for a latch to be released; a latch is a very basic serialization mechanism that can be used to prevent data structures in the SGA being accessed simultaneously.
- Parameters: latch address/latch number/number of sleeps
- Optimization steps:

Refer to SAP Note 767414.

11. **enqueue** (Oracle <= 9i)

enq: <type> - <description> (Oracle >= 10g)

- Meaning: Waiting for an Oracle-Lock
- Parameters: Type/ID1/ID2
- Optimization steps:

Refer to SAP Note 745639

Other Things to Check in Relation to Wait Events

If you see a number of individual wait events that is inexplicably high, it may be due to a CPU bottleneck on the database server. It is therefore conceivable that an Oracle process that has a lock may be displaced by other processes from the CPU. As a result, the period of the lock stopping greatly increases and other processes serialize on this lock. You must therefore use transaction ST06/OS07 to check whether there are sufficient CPU resources. SAP recommends an idle proportion of at least 30% per hour.

Significantly increased average values may occur frequently for individual wait events, due to measurement errors. To prevent the values displayed resulting from individual incorrect statistical values, you can carry out a reset in many SAP screens. The values collected afterward are generally correct since it is unlikely that an incorrect statistical value will appear after a reset.

Appendix: Information provided by the V\$ Views

V\$EVENT_NAME

The V\$EVENT_NAME view shows one row for each wait event known to the Oracle kernel. Associated with each wait event are up to three parameters (additional pieces of information) that provide more detail about a wait situation.



Columns of V\$EVENT_NAME

SQL> DESCRIBE V\$EVENT_NAME

Name	Null?	Type
-----	-----	-----
EVENT#		NUMBER
NAME		VARCHAR2 (64)
PARAMETER1		VARCHAR2 (64)
PARAMETER2		VARCHAR2 (64)
PARAMETER3		VARCHAR2 (64)

V\$SYSTEM_EVENT

The `V$SYSTEM_EVENT` view shows one row for each wait event name, along with the total number of times a process has waited for this event, the number of timeouts, the total amount of time waited, and the average wait time. All of these figures are cumulative for all Oracle processes since the instance started. Wait events that have not occurred at least once since instance startup do not appear in this view.



Columns of `V$SYSTEM_EVENT`

```
SQL> DESCRIBE V$SYSTEM_EVENT
```

Name	Null?	Type

EVENT		VARCHAR2 (64)
TOTAL_WAITS		NUMBER
TOTAL_TIMEOUTS		NUMBER
TIME_WAITED		NUMBER
AVERAGE_WAIT		NUMBER
TIME_WAITED_MICRO		NUMBER

`EVENT` is the name of the wait event.

`TOTAL_WAITS` is the total number of times a process has waited for this event since instance startup. This includes processes from database sessions that have subsequently ended.

`TOTAL_TIMEOUTS` is the total number of times a process encountered a timeout while waiting for an event. When a process begins to wait for an event, it specifies a timeout period after which the operating system should wake it up if the event has not yet transpired. For example, when an Oracle process issues an I/O request to read a block from a data file (the db file sequential read wait event), the process sets a timeout of one second. Usually the I/O request will complete in less than one second and no timeout will occur. But if the read should take longer than one second, a timeout will occur and the process will wake up. The process might do some minor housekeeping, but it will likely just begin another timeout period of one second and continue waiting for the same event.

`TIME_WAITED` and `AVERAGE_WAIT` show the cumulative and average time spent by processes waiting for this event, in centiseconds. Divide these figures by 100 to get the wait time in seconds. These two columns will show as zero if timed statistics are not enabled.

`V$SESSION_EVENT`

The V\$SESSION_EVENT view is similar to the V\$SYSTEM_EVENT view, except that it shows separate rows of information for each Oracle process. Event names do not appear in this view if the process has not waited for them at least once. Also, when an Oracle process terminates (as in the case of when a user logs off the database) all of the rows in V\$SESSION_EVENT for that process permanently disappear.



Columns of V\$SESSION_EVENT

```
SQL> DESCRIBE V$SESSION_EVENT
```

Name	Null?	Type
-----	-----	-----
SID		NUMBER
EVENT		VARCHAR2 (64)
TOTAL_WAITS		NUMBER
TOTAL_TIMEOUTS		NUMBER
TIME_WAITED		NUMBER
AVERAGE_WAIT		NUMBER
MAX_WAIT		NUMBER
TIME_WAITED_MICRO		NUMBER

SID indicates the session ID (not the system ID) of the process waiting for the event. You can query V\$SESSION to determine the SID of the session whose wait events you want to investigate.

The next five columns – EVENT, TOTAL_WAITS, TOTAL_TIMEOUTS, TIME_WAITED, and AVERAGE_WAIT – are the same as in the V\$SYSTEM_EVENT view, except that now they pertain to the one specific process instead of all processes.

MAX_WAIT indicates the maximum amount of time the process had to wait for the event. Like TIME_WAITED and AVERAGE_WAIT, the unit of measure is centiseconds and will display as zero if timed statistics are not enabled.

V\$SESSION_WAIT



Columns of V\$SESSION_WAIT

```
SQL> DESCRIBE V$SESSION_WAIT
```

Name	Null?	Type
-----	-----	-----
SID		NUMBER
SEQ#		NUMBER
EVENT		VARCHAR2 (64)

P1TEXT	VARCHAR2 (64)
P1	NUMBER
P1RAW	RAW (4)
P2TEXT	VARCHAR2 (64)
P2	NUMBER
P2RAW	RAW (4)
P3TEXT	VARCHAR2 (64)
P3	NUMBER
P3RAW	RAW (4)
WAIT_TIME	NUMBER
SECONDS_IN_WAIT	NUMBER
STATE	VARCHAR2 (19)

SID indicates the process.

SEQ# is a sequentially increasing number that starts at one for each process and increments each time the process begins a new wait.

The STATE column indicates how we should interpret the data in this row of the view. If the value in the STATE column is WAITING, then the process is currently waiting for an event. In this case, we can see information about the event and how long the process has been waiting so far. Otherwise, the process is currently not waiting, but we can see information about the last event that the process waited for.

EVENT is the name of a wait event. P1TEXT is the name of a parameter for the wait event, P1 is the value of the parameter, and P1RAW is the value in binary form. The P2 and P3 columns provide additional parameter information.

When the value in the STATE column is WAITING, the value in the WAIT_TIME column will be zero and SECONDS_IN_WAIT will show the number of seconds the process has been waiting for the event thus far. Note that SECONDS_IN_WAIT shows the time in seconds, not centiseconds.

When the value in the STATE column is WAITED KNOWN TIME, WAIT_TIME will show the length of the last wait (in centiseconds) and SECONDS_IN_WAIT will not be relevant (it appears to be the number of seconds since the last wait began, but this is not clear). The STATE could also be WAITED UNKNOWN TIME or WAITED SHORT TIME, the latter indicating that the last wait was less than one centisecond in duration.

Exercise 17: Monitoring Oracle Wait Events

Exercise Objectives

After completing this exercise, you will be able to:

- Analyze the wait events using the System Event monitor in the DBA Cockpit.

Business Example

Your complain about the low performance of your system. To determine where the low performance may originate, you want to analyze the Oracle wait events. You can then determine whether the performance problems are caused by database locks, disk accesses, latches, or another cause.

Task:

Identify the top five time-consuming wait events. Repeat this exercise after you have executed the other exercises in this course and compare the results. You can use the following table to note down the top five wait events.

	Oracle Wait Interface	Monitoring Exclusive Lockwaits	Creating an Index
1			
2			
3			
4			
5			

1. Monitor the Oracle wait events using the DBA Cockpit and identify the top five time-consuming wait events.

Solution 17: Monitoring Oracle Wait Events

Task:

Identify the top five time-consuming wait events. Repeat this exercise after you have executed the other exercises in this course and compare the results. You can use the following table to note down the top five wait events.

	Oracle Wait Interface	Monitoring Exclusive Lockwaits	Creating an Index
1			
2			
3			
4			
5			

1. Monitor the Oracle wait events using the DBA Cockpit and identify the top five time-consuming wait events.
 - a) Call transaction DBACOCKPIT or ST04.
 - b) Switch to the System Events monitor by choosing *Performance* → *Wait Event Analysis* → *System Events*.

The Wait Event Details sub-monitor displays the non-idle wait events. The list is sorted by wait time in descending order.



Lesson Summary

You should now be able to:

- Use Oracle wait event to determine what a session is currently doing or what it is waiting for
- Draw conclusions about the potential for optimization by using the wait events
- Trace the wait events of an Oracle session and determine the most important wait events in previous periods

Lesson: Automatic Workload Repository and Histories

Lesson Overview

This lesson introduces Oracle 10g features for workload analysis, space requirements, and database performance. These new features are enabled by the Automatic Workload Repository.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain the potential of the Automatic Workload Repository to monitor database performance
- Use the DBA Cockpit to analyze the Automatic Workload Repository

Business Example

The Automatic Workload repository provides new features that facilitate workload analysis, space requirements and database performance. You need to know how to use these new features in the SAP environment.

Introduction

As of Oracle 10g, Oracle provides a multitude of new features that facilitate workload analysis, space requirements, and database performance. The database creates a number of new classes of statistics and provides a repository to store these statistics permanently for later analysis. To fulfill these requirements, the Automatic Workload Repository (AWR) was implemented. The data that is collected within the functions that are described here is stored in the SYSAUX tablespace.

The Automatic Workload Repository is source for several functionalities, such as:



- **Active Session History (ASH):** Analysis of historical wait events and CPU consumption of Oracle sessions
- **Automatic Workload Repository (AWR):** Collection of database performance data
- **Automatic Database Diagnostic Monitor (ADDM):** Tuning recommendations based on AWR
- **Segment Advisor:** Information about space requirements and fragmentation of segments
- **SQL Tuning Advisor:** Recommendations about optimizing SQL statements
- **SQL Access Advisor:** Recommendations about optimizing SQL statements

The workload repository enables historical performance analysis by taking snapshots on a periodic basis. The statistic values taken at an AWR snapshot reside on the memory of the database instance. Every time a snapshot occurs, these values are stored in tables located in the SYSAUX tablespace.

AWR snapshots are taken automatically by the database instance from a background process called MMON. AWR snapshots are gathered by default once per hour and they can also be triggered manually, but this is usually not necessary. The length of time in which AWR keeps data around before purging is referred to as the **snapshot retention period**. By default, the retention period is set to seven days. The retention period and the snapshot interval can be changed by the administrator.

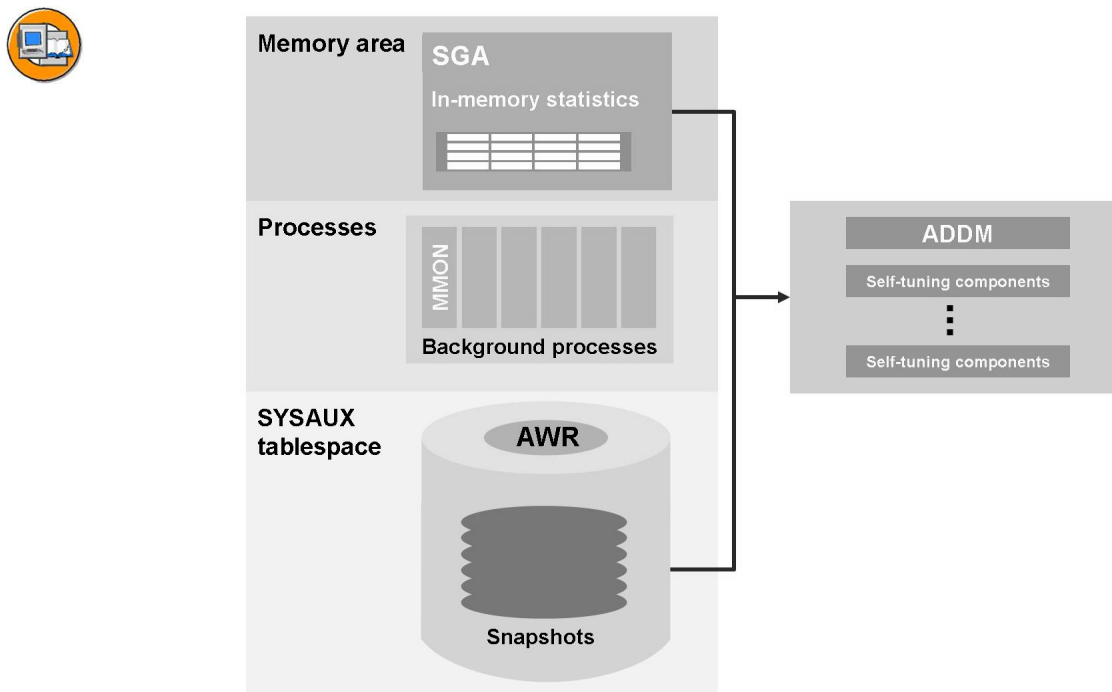


Figure 158: Automatic Workload Repository

Each snapshot taken is assigned a sequence number value as its snapshot ID. Because a single snapshot is not very useful on its own, the sequence is important. Keep in mind that a snapshot is basically a capture of the current values of the database statistics. To determine the statistical changes over a period of time, two snapshots are needed to compute the difference between the snapshots.

AWR collects many classes of statistics during creating a snapshot. The most important classes of statistics are:

Wait event statistics

AWR collects system-level statistics for wait events; this includes the elapsed time for a wait event and the number of occurrences of the wait event.

Active Session History

The Active Session History contains information about the active session of the database instance. ASH data is stored in a circular memory buffer that is captured during an AWR snapshot. The ASH data contains the following attributes: session ID, current SQL statement ID, and current wait event ID.

System statistics

System-wide statistical information, which contains a few hundred general counter, value, and time statistics, is collected during each AWR snapshot.

Top SQL statistics

AWR collects statistics about SQL statements, which can be used to identify expensive SQL statements that have potential for tuning. To reduce the amount of data, AWR collects only the top SQL statements active over the snapshot period. The AWR data contains a variety of information about an SQL statement, for example, the execution plan and information about bind variables.

Top segment statistics

These statistics include information about the segments (tables, indexes, and so on) that are being accessed by the database. This data could be used to identify hot tables, missing indexes, row chaining, and many other data-related issues. To reduce the amount of data, AWR collects only the most active segments over the snapshot period.

Operating system statistics

This data contains information about OS memory usage, CPU utilization, and paging-related information.

Memory statistics

Statistics about the usage of the SGA and the PGA are collected. These statistics give the amount of memory allocated by different database components.

Parameter values

The current values of the database parameters are collected, because parameter values and their changes can have impact on database performance.

Different DBA_HIST views, which allow you to analyze past periods, are available based on the AWR data. The names correspond to the standard performance views to a large extent, the difference being that “V\$” is replaced by “DBA_HIST”_. The following views are especially relevant from a performance point of view:

- DBA_HIST_FILESTATXS: Data file accesses
- DBA_HIST_TEMPSTATXS: Temp file accesses
- DBA_HIST_SYSTEM_EVENT: Accumulated wait events
- DBA_HIST_WAITSTAT: Buffer busy waits
- DBA_HIST_ENQUEUE_STATS: Enqueue waits
- DBA_HIST_LATCH: Latch waits
- DBA_HIST_SYSSTAT: System-wide statistical information
- DBA_HIST_ACTIVE_SESS_HISTORY: Snapshots of the wait event and the CPU consumption of active sessions
- DBA_HIST_SQLBIND: Bind variable content
- DBA_HIST_SQL_PLAN: Execution plans
- DBA_HIST_OSSTAT: CPU information and memory information

These views each contain a SNAP_ID column that uniquely identifies a system snapshot. By default, a snapshot is created every hour.

The Automatic Database Diagnostic Monitor (ADDM) provides system performance analysis based on the AWR data and makes appropriate recommendations for corrective actions. ADDM automatically detects and diagnoses common performance problems, including:

- Hardware issues related to excessive I/O
- CPU bottlenecks
- Connection management issues
- Excessive parsing
- Concurrency issues, such as contention for locks
- PGA, buffer-cache, and log-buffer-sizing issues
- Issues specific to Oracle Real Application Clusters (RAC) deployments, such as global cache hot blocks and objects and interconnect latency issues

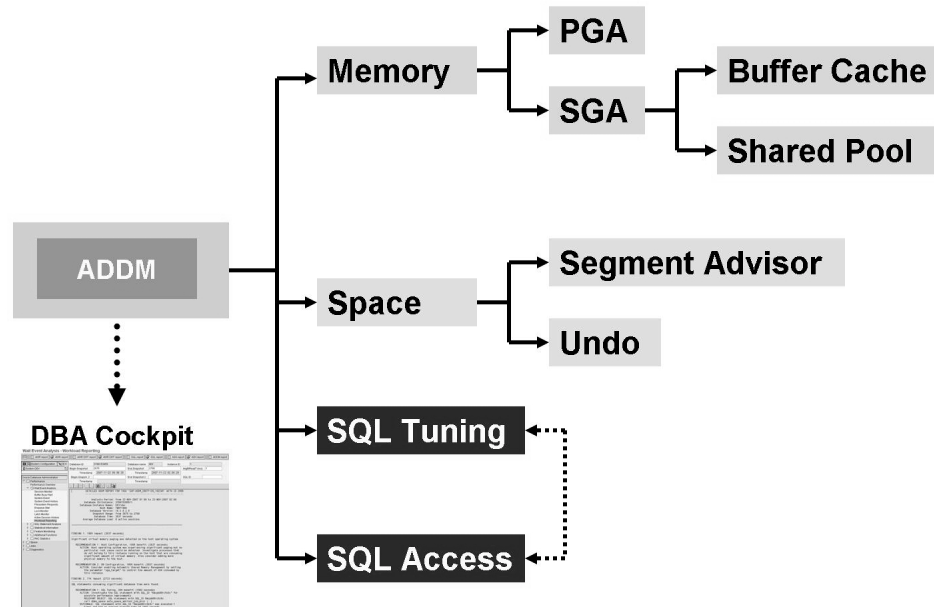


Figure 159: ADDM: Advisory Framework

Workload Reporting Using the DBA Cockpit

The Workload Reporting monitor in the DBA Cockpit lets you display very detailed reports from the Oracle Active Workload Repository. To access this monitor, choose *Performance* → *Wait Event Analysis* → *Workload Reporting* in the DBA Cockpit.

The Workload Reporting monitor offers the following reports for the analysis of workload, space requirements, and database performance.



- Create an AWR overview report
- Create an AWR diff report
- Create an AWR SQL report
- Create an ASH overview report
- Create an ADDM report



Hint: For more information about how to access the information provided by the Automatic Workload Repository without using the DBA Cockpit, see SAP Note 853576. This note describes the analysis of the Automatic Workload Repository using several SQL scripts.

Creating an AWR Overview Report

Within the AWR, snapshots of the system status are created automatically each hour. Based on any two snapshots, you can create a report that gives an overview of the database activities in the period between the snapshots. This report can optionally be displayed in text format or HTML format.



Note: Such an AWR report is an enhancement of the statspack report, which is also available with previous releases (see SAP Note 717484).

Among other things, the AWR report contains the following performance-relevant information:

- Size of the memory areas
- Load values (for example, number of disk reads/seconds)
- Hit ratios
- Non-idle-wait events that have occurred
- Top SQL statements with respect to elapsed time, CPU time, buffer gets, disk reads, number of executions, parses, and sharable memory
- Instance activities (V\$SYSSTAT info)
- Tablespace and data file I/O information
- Buffer busy wait statistics
- Enqueue statistics
- Latch statistics
- Segment statistics
- Profile parameters

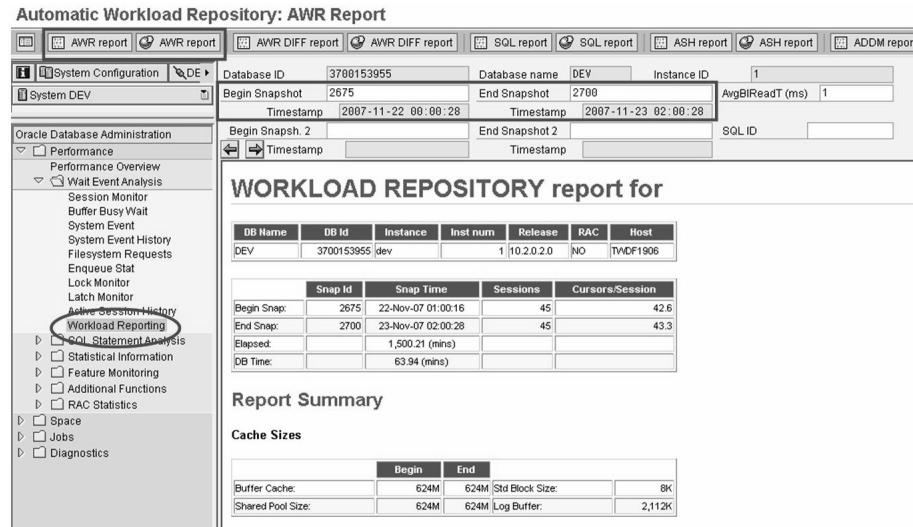


Figure 160: AWR Overview Report

The AWR Overview report in the DBA Cockpit can be created with text output or with Web page display. Therefore you have to choose *AWR report (text)* or *AWR report (web)*. First, you need to enter the parameters **Begin Snapshot** and **End Snapshot**.

Creating an AWR Diff Report

An AWR Diff report allows you to compare AWR data in two different time periods. Such a report is particularly useful if the performance has deteriorated compared to the performance during a past period, even though the processed load is comparable.

The AWR diff report contains the same information as a normal AWR report. However, the system always compares the values of both time periods.

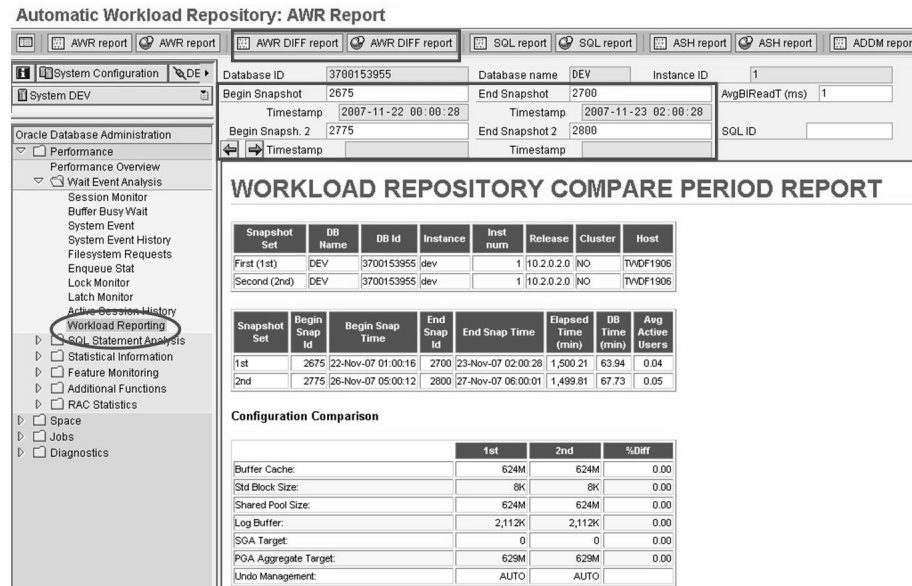


Figure 161: AWR Diff report

The AWR Diff report in the DBA Cockpit could be created with text output or with web page display. Therefore you have to choose *AWR DIFF report (text)* or *AWR DIFF report (web)*. First you need to enter the parameters **Begin Snapshot** and **End Snapshot**. To compare two periods you also have to enter **Begin Snapsh. 2** and **End Snapshot 2** to define the second interval that is to be compared.

Creating an AWR SQL Report

You can create an AWR SQL report for an individual SQL statement. The report contains information such as the SQL statement itself, the load information (for example, disk reads, buffer gets, and elapsed time), and the execution plan.

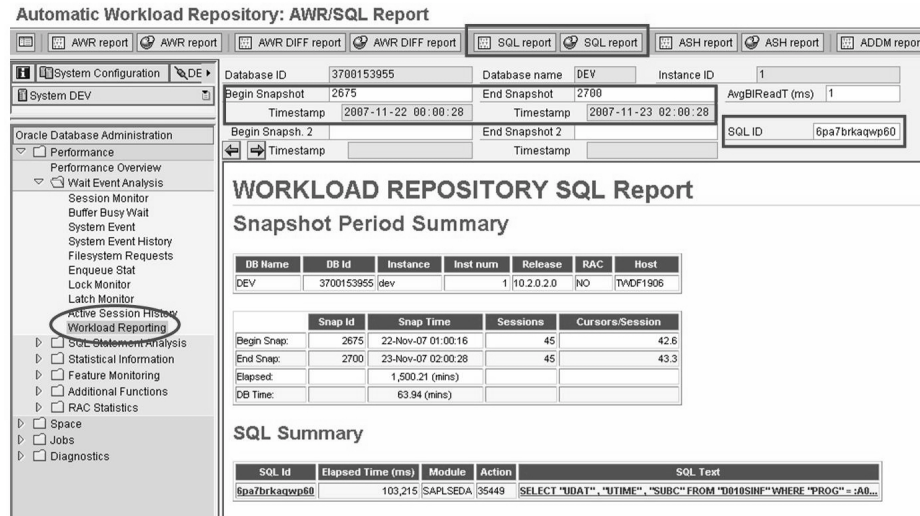


Figure 162: AWR SQL Report

The AWR SQL report in the DBA Cockpit could be created with text output or with Web page display. Therefore you have to choose *SQL report (text)* or *SQL (web)*. First you need to enter the parameters **Begin Snapshot** and **End Snapshot**. Enter the SQL_ID of the SQL statement that is to be analyzed.

Creating an ASH Overview Report

When you run an ASH Overview report for a selected period, you receive the following information:

- Top wait events
- SQL statements for the top activities
- Objects for the top activities
- Top activities for sub-intervals in the selected period

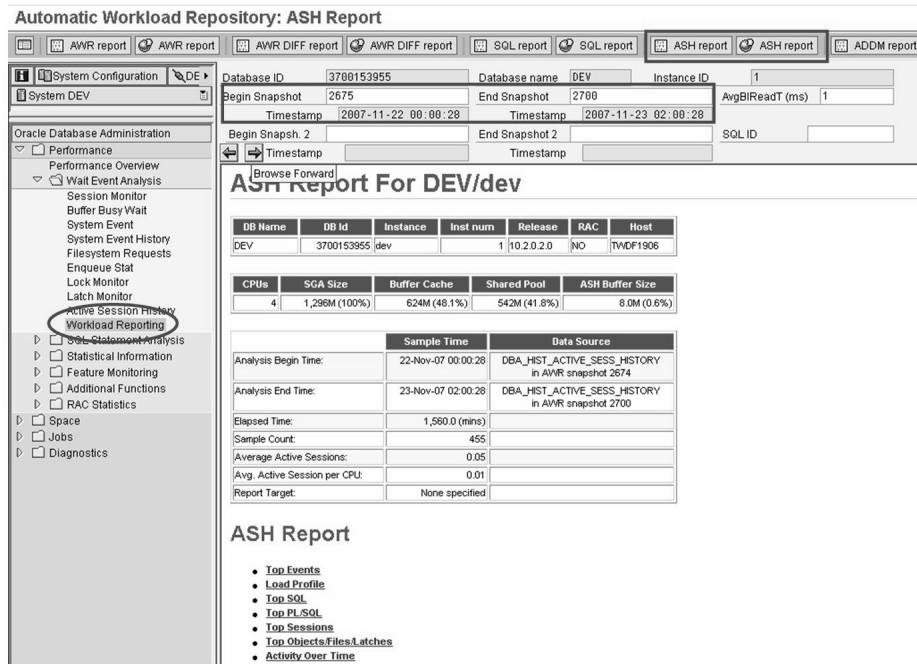


Figure 163: ASH Overview Report

The ASH Overview report in the DBA Cockpit could be created with text output or with Web page display. Therefore you have to choose *ASH report (text)* or *ASH report (web)*. First you need to enter the parameters **Begin Snapshot** and **End Snapshot**.

Creating an ADDM Report

On the basis of the AWR data, you can create an ADDM report that contains automated tuning recommendations and an overview of non-critical areas.

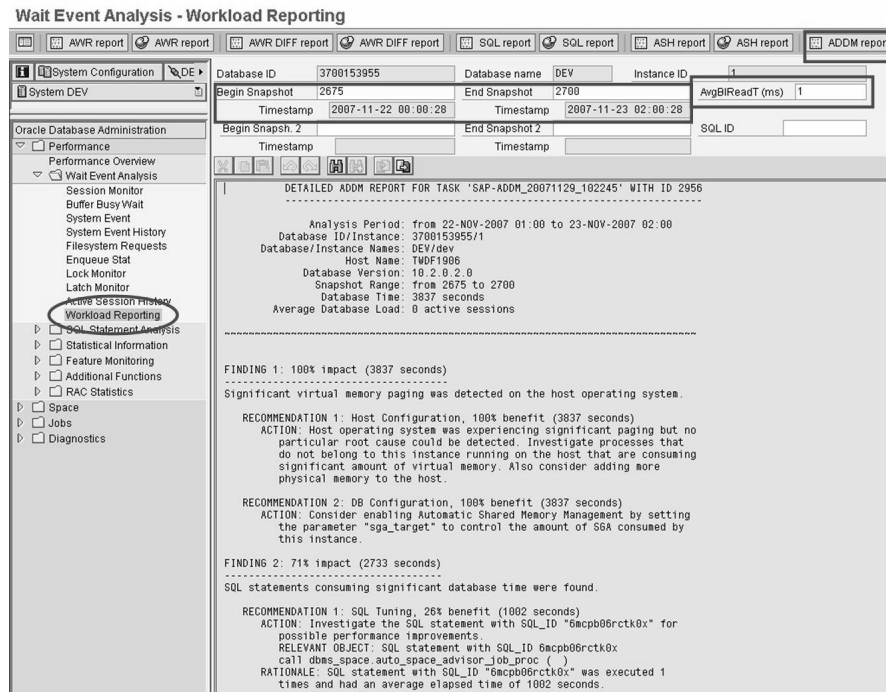


Figure 164: ADDM Report

The ADDM report in the DBA Cockpit could be created with text output. Therefore you have to choose *ADDM report*. First, you need to enter the parameters **Begin Snapshot** and **End Snapshot**. Enter the *average block read time (AvgBlReadT)* in milliseconds for the ADDM report. This is the average time to read a single database block.

Active Session History

The Active Session History monitor in the DBA Cockpit lets you view the Oracle active session history views, which provide sampled session activity in the database instances. To access this monitor, choose *Performance* → *Wait Event Analysis* → *Active Session History* in the DBA Cockpit.

Before the system displays the data, you have to specify the data range in the *Preselect data pool* area:

- Data source

Select the GV\$ACTIVE_SESSION_HISTORY view to display current data or the DBA_ACTIVE_SESSION_HISTORY view to display historical information.

- From and To

Specify a time frame for the data that has to be displayed.

Use the following functions in the *Preselect data pool* area to complete the selection values:

- *Avail. timeframe* gives you information about available data in the two views. This function offers the option to take over this time frame in the selection fields.
- *Set to last hour* fills the time-frame selection with data from the last hour.

Finally, choose *Load data* to read the selected data.

The *ASH List* tab displays the selected data in an ALV list, sorted by time stamp in descending order.



Wait Event Analysis - Active Session History

System Configuration | DE | System DEV

Oracle Database Administration

- Performance Overview
- Wait Event Analysis
 - Session Monitor
 - Buffer Busy Wait
 - System Event
 - System Event History
 - Filesystem Requests
 - Enqueue Stat
 - Lock Monitor
 - Latch Monitor
 - Active Session History
 - Workload Reporting
 - SQL Statement Analysis
 - Statistical Information
 - Feature Monitoring
 - Additional Functions
 - RAC Statistics
 - Space
 - Jobs
 - Diagnostics

Active Session History

DB Name: DEV | Started: 28.11.2007 | DB Server: TWDF1906 | DB Release: 10.2.0.2.0

Preselect data pool

Data source: 1 GV\$ View (GV\$_ACTIVE_SESSION_HISTORY) | From: 25.11.2007 08:46:00 | To: 29.11.2007 09:46:00 | Avail. timeframe | Set to last hour

Load data

ASH List | ASH Graphic

Sample timestamp	Sess ID	Sess State	Event Name	Seq. #	Parameter. 1	P2	P3	WTime [mic]	Waited[mic]
20071129,094158040	150	ON CPU		6	300	0	0	2.999.786	0
20071129,094143041	150	ON CPU		6	300	0	0	2.999.762	0
20071129,094140041	112	ON CPU		34.169	1.413.697.536	1	0	2	0
20071129,094137041	150	ON CPU		6	300	0	0	2.999.799	0
20071129,094131041	150	ON CPU		6	300	0	0	2.999.782	0
20071129,094128041	150	ON CPU		6	300	0	0	2.999.780	0
20071129,094122042	150	ON CPU		6	300	0	0	2.999.779	0
20071129,094120042	112	ON CPU		34.085	1.413.697.536	1	0	1	0
20071129,094116042	150	ON CPU		6	300	0	0	2.999.796	0
20071129,094110042	150	ON CPU		6	300	0	0	2.999.806	0
20071129,094058043	150	ON CPU		6	300	0	0	1.139.876	0
20071129,094055043	112	ON CPU		33.903	1.413.697.536	1	0	28.296.155	0
20071129,093643049	112	ON CPU		33.552	1.413.697.536	1	0	1.430.265	0
20071129,093633049	112	WAITING	db file scattered read	33.428	3	22.772	5	0	8.707
20071129,093530049	112	ON CPU		33.339	1.413.697.536	1	0	4	0
20071129,093026059	113	WAITING	db file sequential read	45.880	15	126.817	1	0	7.635
20071129,092522065	96	WAITING	db file sequential read	47.535	19	15.873	1	0	4.572
20071129,092521065	96	WAITING	db file sequential read	44.977	37	362.318	1	0	5.021

Figure 165: Active Session History: List

The *ASH Graphic* tab displays the selected data in a time chart graphic. Here you have the option to change the graphics display interactively.

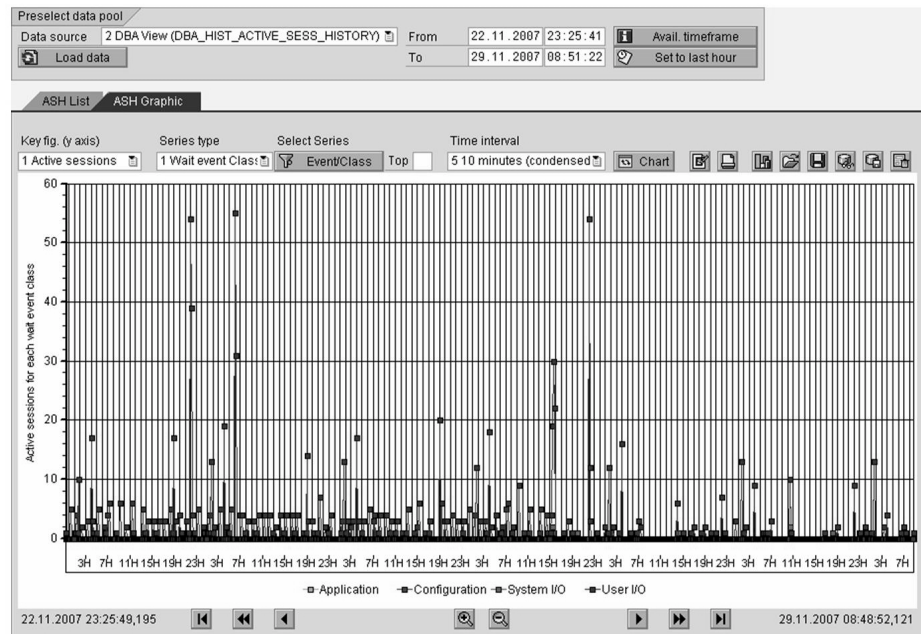


Figure 166: Active Session History: Graph



Lesson Summary

You should now be able to:

- Explain the potential of the Automatic Workload Repository to monitor database performance
- Use the DBA Cockpit to analyze the Automatic Workload Repository



Unit Summary

You should now be able to:

- Use Oracle wait event to determine what a session is currently doing or what it is waiting for
- Draw conclusions about the potential for optimization by using the wait events
- Trace the wait events of an Oracle session and determine the most important wait events in previous periods
- Explain the potential of the Automatic Workload Repository to monitor database performance
- Use the DBA Cockpit to analyze the Automatic Workload Repository



Test Your Knowledge

1. Which of the following wait events are busy wait situations?

Choose the correct answer(s).

- ☐ A CPU used by this session
- ☐ B log file sync
- ☐ C db file sequential read
- ☐ D SQL*Net message from client

2. Which of the following wait events may be caused by full table scans?

Choose the correct answer(s).

- ☐ A enqueue
- ☐ B log file parallel write
- ☐ C db file scattered read
- ☐ D log buffer space



Answers

1. Which of the following wait events are busy wait situations?

Answer: B, C

Only B (log file sync) and C (db file sequential read) are busy wait situations. SQL*Net message from client is an idle wait event and the CPU time is not covered by the Oracle Wait Interface.

2. Which of the following wait events may be caused by full table scans?

Answer: C

db file scattered read may be caused by full table scans.

Unit 8

Analyzing application design

Unit Overview

This unit describes the impact of expensive SQL statements on the performance of your system and introduces the tools provided by SAP to detect expensive SQL statements.



Hint: In this unit only expensive SQL statements resulting from requests within the ABAP -side of the SAP system are covered.



Note: You will find many screenshots in this unit. Please note that sometimes it had been necessary to change the display settings (from the initial settings) to get useful screenshots.



Unit Objectives

After completing this unit, you will be able to:

- Explain why even a few expensive SQL statements may reduce performance for the entire SAP system
- Use the work process overview to find potentially expensive SQL statements currently executed in your SAP system
- Use workload analysis to identify transactions in SAP systems that bring significant load to the database
- Use statistical records to identify transactions in SAP systems that bring significant load to the database
- Use the database performance monitor to find expensive SQL statements
- Identify SQL statements that you can tune
- Use the SQL trace to analyze the processing of SQL statements in detail on database level
- Use the Explain function
- Explain when a lock wait situation occurs and how you can reduce lock waits

- Use the Lock Monitor to identify exclusive lock wait situations

Unit Contents

Lesson: Consequences of Expensive SQL Statements	483
Lesson: Using SM50/SM66 to Find Expensive SQL Statements	487
Lesson: Using ST03/STAD to Find Expensive SQL Statements	490
Lesson: Using ST04 to Find Expensive SQL Statements	494
Exercise 18: Detecting Expensive SQL Statements Using the DBA Cockpit	503
Lesson: Using the SQL Trace to Find Expensive SQL Statements	508
Lesson: Exclusive Lock Waits	516
Exercise 19: Monitoring Exclusive Lock Waits	529

Lesson: Consequences of Expensive SQL Statements

Lesson Overview

In this lesson you will learn about the impact of expensive SQL statements on the performance of your database or SAP system.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain why even a few expensive SQL statements may reduce performance for the entire SAP system

Business Example

You have detected performance problems in your SAP system that are caused by expensive SQL statements in some applications. To find expensive SQL statements, you want to use different tools in the SAP system.

Introduction to Expensive SQL Statements

Expensive SQL statements (**expensive statements** or **expensive selects**) might hide in all kinds of requests to the database. Not all expensive statements on the database are caused by badly written code; they may also result from inappropriate use of functions of the SAP system.

Expensive statements are defined as all SQL statements that cause the database to read many blocks (from disk or buffer).



Note: It is not so easy to quantify “many blocks” in the definition of expensive SQL statements. Usually statements are called **expensive** if the database reads more than 5% of the total blocks it reads for the answering of those statements (according to the Performance Overview monitor of DBA Cockpit).

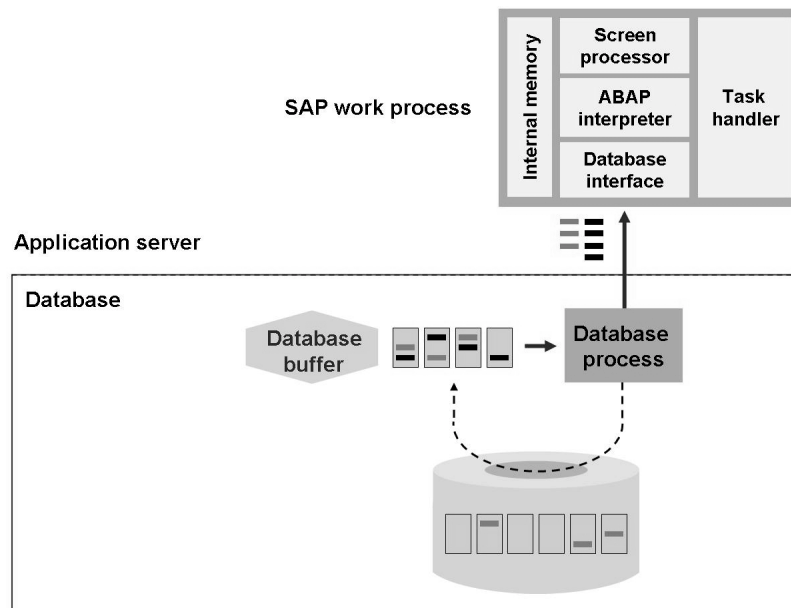


Figure 167: Consequences of Expensive SQL Statements

Expensive SQL statements might have negative consequences on (global) SAP system performance:



- The database is busy reading many blocks. All database tasks might be slowed down; this may impact the overall performance of the SAP system.
- Many blocks might be displaced from the database buffer; this can have a negative impact on following requests.
- The work process is blocked (waiting for the database response) and is therefore not available for other requests (potentially increasing their wait time).

As you can see from the wording above (“might,” “may,” “potentially,” and “can”), there is no definitive (and always correct) rule for the impact of expensive SQL statements on system performance. Whether the expensive statement has a negative impact on the system's performance depends on the following criteria:



- How often is the statement executed?
- When is the statement executed?
- What is the overall impact on the system's total response time?

Obviously an expensive statement has greater relevance when it is executed, for example, several thousand times per day, compared to a statement executed only once per year. It is usually better to have a statement that executes once a year and takes five hours to get a result than to have a statement that executes 10,000 times per day and takes 10 seconds to get results.

An expensive statement executed several times during non-working hours in the background has less impact as the same number of executions done in dialog on Monday morning.

Using the workload analysis (transaction ST03) you can sum up, for example, the database request times for all transactions for the day. Assuming that this number is around 100,000, the single execution of an expensive statement contributing 10 seconds to the overall database request time during the day is not a top priority from the point of tuning. Such a statement might be an expensive statement, but might not be worth tuning because of insignificant impact.

➔ **Note:** Not all expensive SQL statements are worth tuning!

SAP systems offer several tools for finding expensive SQL statements. The following table lists the most important questions related to expensive statements and the tools you can use to answer these questions.



Expensive SQL Statements: Important Questions and Where to Find the Answers in SAP Systems

Question	Where to find the answer
1. Which reports/transactions contain expensive selects?	SM50/SM66, ST03, STAD
2. Which table is accessed expensively?	SM50/SM66, ST04/DBA-COCKPIT
3. Which index is used for the access?	ST04/DBACOCKPIT, ST05
4. Which WHERE clause is being used?	ST04/DBACOCKPIT, ST05, SM50
5. Which statement is worth tuning?	ST03, ST04/DBACOCKPIT

➔ **Note:** For SAP systems based on SAP Web AS 6.40, you can use transaction ST04 instead of the DBA Cockpit.



Lesson Summary

You should now be able to:

- Explain why even a few expensive SQL statements may reduce performance for the entire SAP system

Lesson: Using SM50/SM66 to Find Expensive SQL Statements

Lesson Overview

In this lesson you will learn how to use the local and global work process overviews (transactions SM50 and SM66) to find expensive SQL statements currently executed in your SAP system.



Lesson Objectives

After completing this lesson, you will be able to:

- Use the work process overview to find potentially expensive SQL statements currently executed in your SAP system

Business Example

The production SAP system at your company currently shows very bad response time. You would like to check if expensive SQL statements are causing the problem.

Detecting Expensive SQL Statements Using the Work Process Overview

The local and global work process overview is a very valuable tool for analyzing the current system state. However, it is worthless for analyzing historical situations (for example, yesterday morning).



Global Work Process Overview

Sort: Server

Server Name	No.	Type	PID	Status	Reason	Sm	Start	Error	CPU	Time	User Names	Report	Action	Table
twdf1906_DEV_00	0	DIA	7972	Running			Yes	2	1	9	SAP_PERF035	SAPMSYST		
twdf1906_DEV_00	1	DIA	3556	Running			Yes	2	3		SAP_PERF051	SAPLSNR3	Direct Read	NRIV
twdf1906_DEV_00	2	DIA	6712	Running			Yes	2	3		SAP_PERF072	SAPLV60A	Insert	VBDATA
twdf1906_DEV_00	4	DIA	6244	Running			Yes	2	3	3	SAP_PERF038	SAPMSYST		
twdf1906_DEV_00	5	DIA	7044	Running			Yes	2	6	37	SAP_PERF031	SAPMSYST		
twdf1906_DEV_00	6	DIA	240	Running			Yes	2	1	7	SAP_PERF037	SAPMSYST		
twdf1906_DEV_00	7	DIA	5140	Running			Yes	3	2	7	SAP_PERF036	SAPMSYST		
twdf1906_DEV_00	8	DIA	4896	Running			Yes	2	1	27	SAP_PERF033	SAPMSYST		
twdf1906_DEV_00	9	DIA	2464	Running			Yes	2	5	26	SAP_PERF034	SAPMSYST		
twdf1906_DEV_00	10	DIA	4680	Running			Yes	3	1	28	SAP_PERF032	SAPMSYST		
twdf1906_DEV_00	11	DIA	6392	Running			Yes	1	3		SAP_PERF082	SAPLSNR3	Direct Read	NRIV
twdf1906_DEV_00	12	DIA	5288	Running			Yes	2	1	51	SAP_PERF030	SAPMSYST		
twdf1906_DEV_00	13	DIA	2076	Running			Yes	2	13		SAP_PERF073	SAPMV60A		
twdf1906_DEV_00	14	UPD	3724	Running			Yes		77		SAP_PERF097	RSM13000	Commit	
twdf1906_DEV_00	15	UPD	3612	Running			Yes		49		SAP_PERF012	RSM13000	Commit	
twdf1906_DEV_00	16	UPD	4012	Running			Yes		34		SAP_PERF018	SAPLV60U	Insert	VBRP
twdf1906_DEV_00	22	BTC	4068	Running			Yes		36	251	DDIC	SAPLSD00	Sequential Read	dba_segmen
twdf1906_DEV_00	28	UP2	2084	Running			Yes		33		SAP_PERF087	RSM13000	Commit	

Figure 168: Detection Using the Global Work Process Overview

How to use the global work process overview:



- Identify long-running database-related actions (for example, Sequential Read, Insert, or Update).
- Note down the report name and, in transaction SM66, the transaction name for later detailed analysis.
- Note down the table name against which the action is running.
- Remember the name of the user executing the long-running transaction; later you will be able to ask that user for assistance for an SQL trace recording (ST05).
- In transaction SM50, you can also see the currently executed SQL statement by double-clicking a line.



Lesson Summary

You should now be able to:

- Use the work process overview to find potentially expensive SQL statements currently executed in your SAP system

Related Information

- Use the F1 help on the *Action* field in transaction SM50 to see a list of the possible actions.

Lesson: Using ST03/STAD to Find Expensive SQL Statements

Lesson Overview

Workload analysis and statistical records offer excellent ways to find functions in SAP systems that bring heavy load to the database. You will learn how to use these tools with respect to the identification of statements that might be tuned.



Lesson Objectives

After completing this lesson, you will be able to:

- Use workload analysis to identify transactions in SAP systems that bring significant load to the database
- Use statistical records to identify transactions in SAP systems that bring significant load to the database

Business Example

Yesterday morning there was a time window when the response time in your SAP system was very bad. You would like to take a closer look at the reason. You would like to check if expensive SQL statements against the database are to blame.

Detection of Expensive SQL Statements Using the Transaction Profile and Statistical Records

First steps in ST03

Using the transaction ST03 transaction profile, you can identify transactions that cause a significant part of total response time, as well as transactions with high database request times – another indicator for expensive SQL statements.

Choose *Standard Transaction Profile* in transaction ST03 and restrict the display to task type *Dialog* (this restriction is optional and sometimes not wanted).

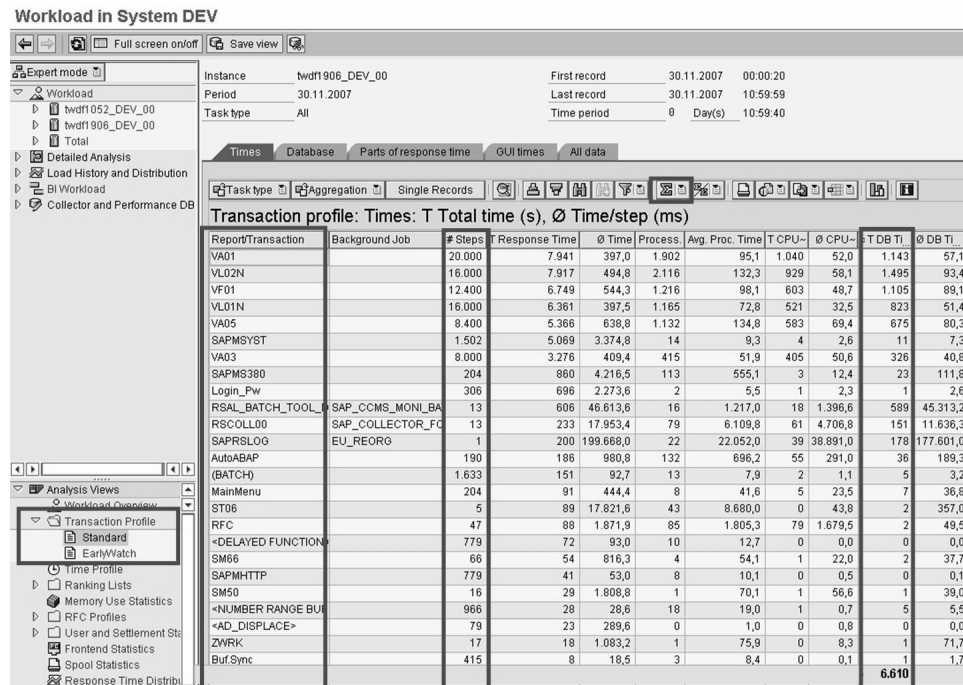


Figure 169: Use the Transaction Profile to Find Expensive SQL Statements

How to use the transaction profile:



- Sort the *Total Database Time* column in ascending order and sum it up. Transactions that are causing more than 5% of the summed (total) database time might be worth tuning.
- It might also be useful to sort the *Average Database Time per dialog step* column to find transactions with high average load on the database. However, most of the transactions found this way are **not** worth tuning because of insignificant overall impact.

The transactions found by this procedure are likely to contain expensive statements.



Note: Some background information:

In the figure above, VA01 has been slowed down by a Z* program attached to it via a **customer exit**. This program, ZZSELECT, executes an expensive select on table ZLIPS.

First steps in STAD

Enter transaction STAD and restrict the display to the dialog steps taking more than, for example, 1000 ms of database request time. This threshold depends on your requirements.

How to use statistical records:



- On the STAD transaction entry screen, enter the following restrictions:
 1. Choose a time frame (remember that statistical records are only available for a certain time).
 2. Select *Task Type D* (for example).
 3. Enter a relevant value for *DB request time*, for example, **1000 ms**.



SAP Workload: Single Statistical Records - Overview

Download Disp. mode Sel. fields

System: DEV Number of RFCs which responded (without errors): 1 (1)

Analysis time: 30.11.2007 / 11:15:00 30.11.2007 / 11:35:00

Display mode: All statistic records, sorted by time

Started	Server	Transaction	Program	T	Scr.	Wp	User	Response time (ms)	Time In VFs (ms)	Wait time (ms)	CPU time (ms)	DB req. time (ms)
		*	*	*	*			0			0	0
11:14:59	twdf1986_DEV_00	V0A01	SAPWV45A	0	0181	0	SAP_PERF000	6.235	6.235	0	125	2.408
11:14:59	twdf1986_DEV_00	V0A01	SAPWV45A	0	0181	1	SAP_PERF001	6.204	6.204	0	156	2.478
11:15:00	twdf1986_DEV_00		MainMenu	0	0040	2	SAP_PERF004	63	63	0	31	25
11:15:00	twdf1986_DEV_00	V0A01	SAPWV45A	0	0181	2	SAP_PERF002	4.438	4.438	0	63	1.926
11:15:01	twdf1986_DEV_00		Login_Pw	0	0020	3	UNKNOWN	48	48	0	0	1
11:15:01	twdf1986_DEV_00	V0A05	SAPWV75A	0	0500	4	SAP_PERF003	302	300	0	156	31
11:15:02	twdf1986_DEV_00	V0A05	SAPWV75A	0	0180	3	SAP_PERF004	42	42	0	16	6
11:15:03	twdf1986_DEV_00		MainMenu	0	0040	3	SAP_PERF005	90	90	0	16	22
11:15:04	twdf1986_DEV_00	V0A05	SAPWV75A	0	0500	3	SAP_PERF004	507	505	0	94	32
11:15:04	twdf1986_DEV_00		Login_Pw	0	0020	3	UNKNOWN	4	4	0	0	1
11:15:04	twdf1986_DEV_00	V0A05	SAPWV45A	0	0181	4	SAP_PERF003	1.355	1.355	0	63	672
11:15:05	twdf1986_DEV_00	V0A01	SAPWV75A	0	0180	3	SAP_PERF005	10	10	0	0	5
11:15:06	twdf1986_DEV_00		MainMenu	0	0040	3	SAP_PERF006	41	41	0	16	22
11:15:07	twdf1986_DEV_00	V0A01	SAPWV45A	0	0480	1	SAP_PERF008	6.073	6.073	0	78	2.766
11:15:07	twdf1986_DEV_00	V0A01	SAPWV45A	0	0480	1	SAP_PERF001	6.087	6.087	0	125	2.766
11:15:07	twdf1986_DEV_00	V0A01	SAPWV45A	0	0480	2	SAP_PERF002	6.084	6.084	0	100	2.720
11:15:07	twdf1986_DEV_00	V0A05	SAPWV75A	0	0500	3	SAP_PERF005	341	339	0	141	26
11:15:07	twdf1986_DEV_00	V0A01	SAPWV75A	0	0181	3	SAP_PERF004	19	19	0	16	1
11:15:07	twdf1986_DEV_00		Login_Pw	0	0020	3	UNKNOWN	4	4	0	0	1
11:15:07	twdf1986_DEV_00	V0A01	SAPWV75A	0	0480	4	SAP_PERF003	6.114	6.114	0	180	2.822
11:15:08	twdf1986_DEV_00		SAPPHHTTP /sap/public/cman	H	3		SAPSYS	12	12	0	0	0
11:15:08	twdf1986_DEV_00	V0A05	SAPWV75A	0	0180	3	SAP_PERF006	15	15	0	16	16
11:15:08	twdf1986_DEV_00		(BAT01)	9	2		SAPSYS	16	16	0	16	2
11:15:09	twdf1986_DEV_00	V0A01	SAPWV45A	0	0480	3	SAP_PERF004	4.180	4.180	0	109	1.716

Figure 170: Use Statistical Records to Find Expensive SQL Statements

You can find a lot of useful information on the resulting screen:

- What transaction caused the high database load?
- What program caused the high database load?
- Which user called the expensive statements?
- How long did it take the database to answer the request?

In the *Analysis of ABAP/4 database requests* section, you can find information on what type of database activity was responsible for the high response time (read activity, write activity, and so on) by double-clicking one of the lines in STAD.



Lesson Summary

You should now be able to:

- Use workload analysis to identify transactions in SAP systems that bring significant load to the database
- Use statistical records to identify transactions in SAP systems that bring significant load to the database

Related Information

- You can find more information on these functions in the documentation by searching for the terms “ST03” and “STAD.”

Lesson: Using ST04 to Find Expensive SQL Statements

Lesson Overview

The database performance monitor (ST04/DBACOCKPIT) offers very detailed information on the work of the database system. In this lesson, you will learn how to use selected functions in the database monitor with respect to finding expensive SQL statements.



Lesson Objectives

After completing this lesson, you will be able to:

- Use the database performance monitor to find expensive SQL statements
- Identify SQL statements that you can tune

Business Example

Your productive SAP system is slowed down because of expensive SQL statements. You are determined to find those expensive SQL statements using some functions offered by the database monitor.

Using the Database Performance Monitor

The database performance monitor (transaction ST04/DBACOCKPIT) is a very valuable tool that is implemented in a different way for each SAP-supported database system. We will focus on ST04 as implemented for the Oracle database.

Transaction ST04 offers a direct view into the current database processes. This is helpful if you encounter a currently long-running transaction (for example, in the global work process monitor) and you would like to know in detail what is happening.



Analyze DB Performance: Oracle Session

Session Monitor

DB Name: DEV Started: 28.11.2007
DB Server: TWDF1906
DB Release: 10.2.0.2.0

Oracle Database Administration

- Performance
 - Performance Overview
 - Wait Event Analysis
 - Session Monitor**
 - Buffer Busy Wait
 - System Event
 - System Event History
 - Filesystem Requests
 - Enqueue Stat
 - Lock Monitor
 - Latch Monitor
 - Active Session History
 - Workload Reporting
- SQL Statement Analysis
- Statistical Information
- Feature Monitoring
- Additional Functions
- RAC Statistics
- Space
- Jobs
- Diagnostics

Analyze since DB start.

Oracle	Client syst	Client	Status	Event	SQL Statement	Logical R	Phys. Re	Block Ch
84 9012	TMPWORKIT	8516:85	ACTIVE	db file sequ	SELECT * FROM "ZMARA" WHEF	216052	8805	42240
86 5692	TMPWORKIT	8456:85	INACTIVE	SQL*Net mes		263694	5457	31603
89 6604	TMPWORKIT	8488:85	ACTIVE	SQL*Net mes	SELECT T1.INST_ID, T1.SID, T1	104263	452	15218
92 5216	TMPWORKIT	8640:85	INACTIVE	SQL*Net mes		26	0	0
93 3612	TMPWORKIT	8444:84	INACTIVE	SQL*Net mes		154510	1578	21570
94 7936	TMPWORKIT			mes		54865	359	28949
95 6036	TMPWORKIT			mes		258300	1571	153058
96 4044	TMPWORKIT	6070:80	INACTIVE	SQL*Net mes		17062	1429	588
98 4628	TMPWORKIT	8500:81	INACTIVE	SQL*Net mes		331911	2780	77401
99 6660	TMPWORKIT	8544:85	INACTIVE	SQL*Net mes		160491	27881	16984
100 8940	TMPWORKIT	8532:85	INACTIVE	SQL*Net mes		103146	447	14288
101 2416	TMPWORKIT	8440:84	INACTIVE	SQL*Net mes		153060	1187	21928
102 6016	TMPWORKIT	8508:84	INACTIVE	SQL*Net mes		166998	1427	23339

Click

Figure 171: The Oracle Session Monitor

How to use the Oracle Session Monitor:

1. Enter transaction ST04 and then choose *Performance* → *Wait Event Analysis* → *Session Monitor*.
2. Identify the long-running task by finding the corresponding work process ID. Compare the first value in the *Client Proc.* column with the process ID of a work process in SM50/SM66.
3. Check the current action on the database by clicking the statement in the *SQL text* column.

In the resulting dialog box, you will find the option to *Explain SQL statement*.

4. Analyze the Explain.

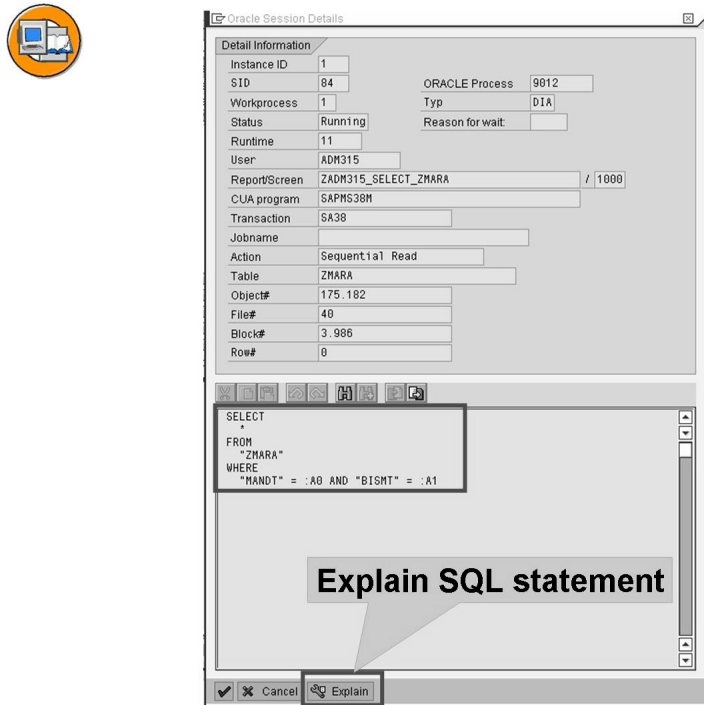


Figure 172: Calling the Explain Function from the Oracle Session Monitor

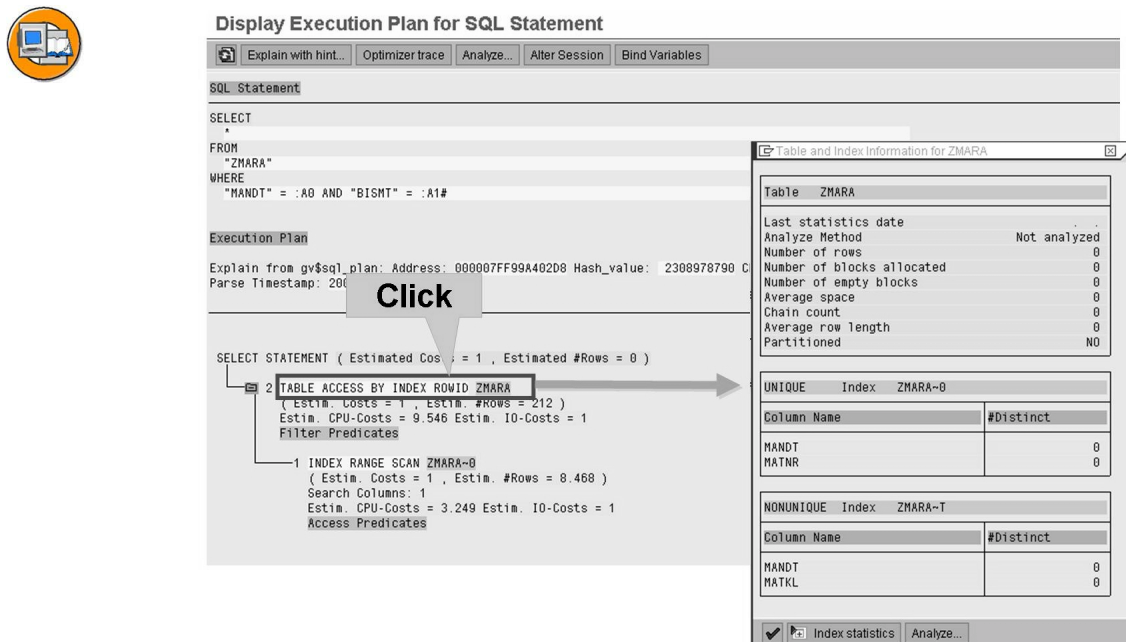


Figure 173: The Explain Function

Monitoring (Database) Buffer Gets

In the figure below, three records are transferred to the work process. All three records were found in the database buffer; however, one of the records – the one shown in red – was read from disk before it could be found in the buffer. To find the red record, three blocks on disk needed to be read.

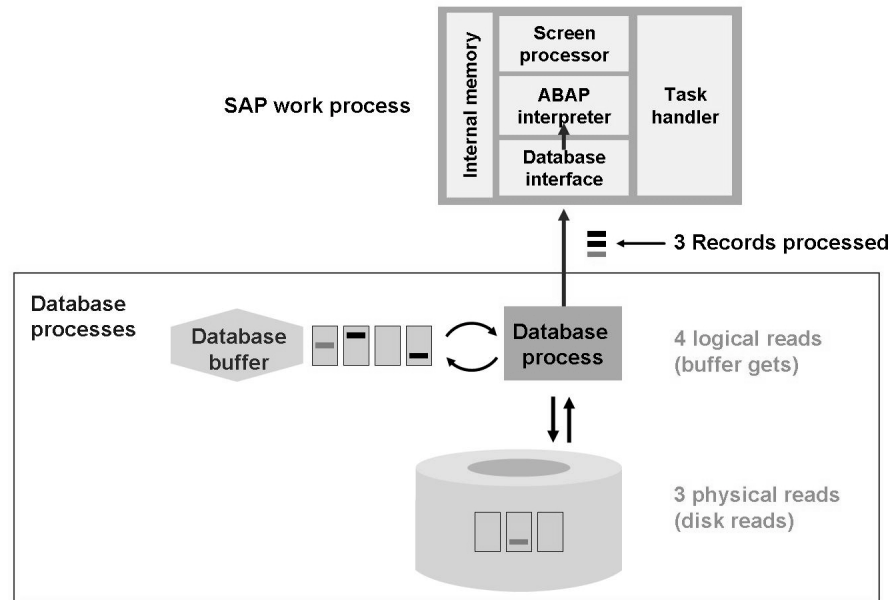


Figure 174: Logical Reads and Buffer Gets

While monitoring the load on the database, you might encounter several different terms denoting similar or identical processes.



Caution: There might be some confusion while encountering the following terms:

Logical reads

Number of Oracle buffer blocks read for the statement from the data buffer (summary found on the entry screen of transaction ST04).

All logical reads from the Oracle database are in reads from the data buffer. To provide the data in the data buffer (before access to the buffer is successful), a physical read must sometimes be executed. Therefore, each physically read block possesses a complementary block read from the buffer. However, most blocks read from the buffer do NOT have a complementary read from disk (except the very first access).

Buffer gets

Number of Oracle buffer blocks read for the statement from the data buffer. Identical in meaning to logical reads.

Physical reads/disk reads

Number of Oracle blocks read for the statement from the hard disk (summary found on the entry screen of transaction ST04 and in *Performance → SQL Statement Analysis → Shared Cursor Cache: Analysis of Shared Cursor Cache*).

You can find information on buffer gets by choosing *Performance → SQL Statement Analysis → Shared Cursor Cache* in the DBA Cockpit (transaction ST04 / DBACOCKPIT).

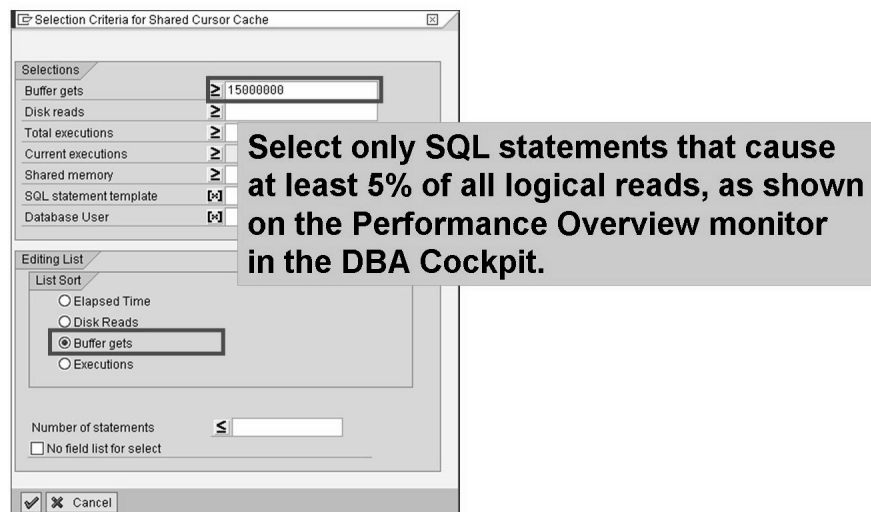


Figure 175: Selecting SQL Statements Concerning Database Load

The resulting screen offers a lot of information and a very important function.

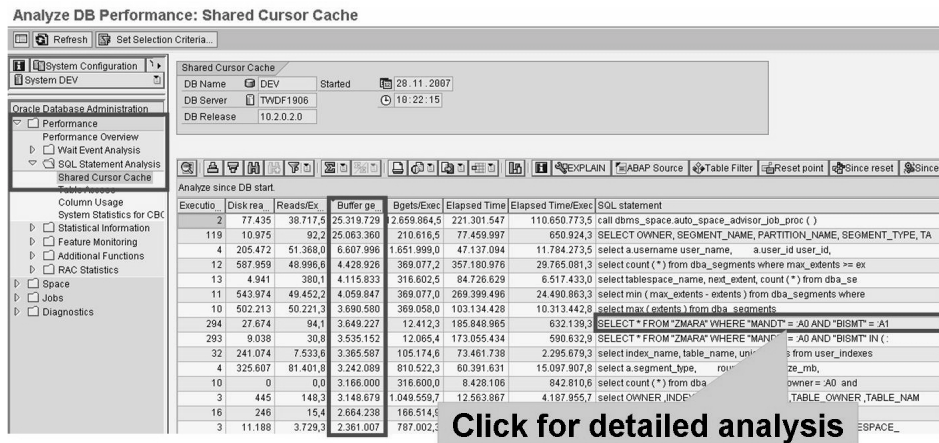


Figure 176: Detailed Analysis of Expensive SQL Statements

Using the following procedure, you will be able to identify expensive SQL statements that have significant impact on the performance of your database system. You will also learn what type of statements you can tune, and how.

Finding expensive SQL statements using the shared cursor cache:



1. Call ST04 and choose *Performance* → *SQL Statement Analysis* → *Shared Cursor Cache: Analysis of Shared Cursor Cache*.
2. On the selection screen, enter the following:

For *Buffer gets*, choose a number equal to 5% of logical reads from the ST04 entry screen. There might be expensive statements causing less than 5% load on the database, but these usually have little impact on system performance, even when optimized.

Choose *List sort* for *Buffer gets*.

3. On the resulting screen, you can analyze the SQL statements that are causing the highest load on your database system.

In the *SQL statement* column, you will find the statement causing the buffer gets.

4. Click on the SQL statement and proceed to analyze this state by using the Explain function.



Note: Note the very important function offered here, called **Display call point in ABAP program**.

You have identified and analyzed the SQL statements that cause the highest number of buffer gets on your database system. These statements are likely candidates for tuning measures. However, not all SQL statements can be optimized by you. Also,

you need to know that you have not yet found the statements causing the highest number of hard disk accesses – we did not select for those. If you are interested in those statements, change the selection options accordingly.

Different Types of Expensive SQL Statements

You are not able to tune all SQL statements that you can see in the shared cursor cache.

SQL statements used by ABAP programs

SQL statements used by ABAP programs are displayed in uppercase letters and quotation marks. These statements can be tuned.

SQL statements used by database administration tools

SQL statements used by database administration tools are displayed in upper case without quotation marks and cannot be tuned by you. These statements are generated by database administration tools that (for example) are called periodically by report RSCOLL00 according to table TCOLL. If such tools cause load problems on your database system, try to schedule these tools to run less frequently. Conduct a search in the SAP Notes database using the name of an offending tool as a search string.

SQL statements selected from SAP Basis tables

These statements are identified by their accesses to SAP Basis tables, such as DDNTT, DDNTF, D010L, and D010INF. These statements cannot be tuned by you. If these statements cause load problems on your database, you should check the SAP buffers by using transaction ST02. The content of the tables named above (among many other Basis tables) should be buffered on application level, thus avoiding unnecessary database calls. Increased read accesses on these tables might indicate a buffering problem.

Recursive SQL statements (Oracle)

Recursive statements are displayed in lowercase letters and cannot be tuned by you. This type of statement is executed, for example, when the database needs to refer to meta-information for fulfilling another SQL request. An example of meta-information is the dictionary information in the database system. Another source for recursive statements is internal activities of the Oracle database (“housekeeping”), like those needed for database space management.



Hint: There are some very helpful selection criteria other than the sort for the highest number of buffer gets.

For example you might choose to restrict the types of SQL statements displayed to the statements that can be tuned by you, by using the *SQL statement template* field with the restriction *****.



Note: Please also consider sorting for the following criteria:

1. Buffer gets/execution
Considers rarely called, but extremely expensive, statements (typically batch jobs)
2. Disk reads
Expensive statements concerning I/O performance
3. Buffer gets/record
Expensive statements for database and application server performance
4. Records/execution
For example, many buffer gets for many records, but also often executed – might be a good statement



Note: Another option for finding (potentially) expensive SQL statements considers the execution plans created by the database. For example it is possible to search for full table scans using the following method:

- Enter transaction ST04.
- Choose *Performance* → *Additional Functions* → *GV\$-Views*.
- Double-click on the line *V\$SQL_PLAN* under *..V\$Views* header (the right-hand column).
- Right-click on the *Operation* column header and choose *Set Filter*.
- Select for the operation TABLE ACCESS and choose *Enter*.
- Repeat the previous step for the *Options* column with the value *Full*.
- Right-click the *OBJECT_NAME* column header and choose *Sort in descend order*.

The result will be a list of tables accessed by full table scans, sorted with Z* tables on the top. These table accesses might be tuned.

Other filter options might be useful as well.



Scenario 1: The optimizer chooses an unsuitable access path
Symptom: Many buffer gets, but only a few records per execution

This type of SQL statement may be accelerated by:

- Updating the optimizer statistics
- Creating an index
- Extending an existing index
- Dropping an existing index
- Adapting the ABAP code
- Optimizing user input

Figure 177: Tunable SQL Statements



Scenario 2: The optimizer chooses a suitable access path
Symptom: Many buffer gets and many records per execution

This type of SQL statement may be accelerated by:

- Adapting the ABAP code
 - Especially if the statement contains a
“SELECT * FROM ...”,
replace the “*” with the list of fields that are really used by the program.
- Tuning the business process
- Optimizing user input

Figure 178: Tunable SQL Statements (2)

You should now be able to find expensive SQL statements using the powerful functions of the database Performance Monitor (transaction ST04) and make recommendations on how to tune them.

Exercise 18: Detecting Expensive SQL Statements Using the DBA Cockpit

Exercise Objectives

After completing this exercise, you will be able to:

- Detect expensive SQL statements using some of the functions offered by transaction ST04

Business Example

Your company's productive SAP system runs with bad performance. After finding suspicious database activities using transactions SM50 and ST03, you would like to do a more detailed analysis using the functions of the database monitor (transaction ST04).

Task:

Analyze an expensive SQL statement using some functions of transaction ST04.



Caution: This exercise is connected with several other exercises in the ADM506 course. These exercises include the exercise on using the SQL Trace (ST05), the exercise on table statistics, and the exercise on the cost-based optimizer.

Therefore the scope of this introductory exercise might seem somewhat limited.

1.

Execute report **ZZSELECT_##** (where **##** stands for your group number, assigned to you by your instructor) several times, using transaction SE38.

Check the source code of the report you are executing using transaction SE38 and identify the name of the table accessed and the names of the fields selected.

2. Analyze the table design of the table accessed by your report, **ZZSELECT_##**, concerning existing indexes. Use transaction SE11 to answer this question.



Note: Be aware of the fact that index design and creation is covered in another lesson. It is sufficient, for this exercise, to check the existing index(es).

Continued on next page

3. Analyze the Oracle Session Monitor and look for table accesses generated by the reports **ZZSELECT_##**. To see some entries in the Oracle Session Monitor, re-execute the report assigned to your group several times.

By double-clicking the corresponding entry in the *SQL text* column, you find some information on the statement executed.

4. Analyze the shared SQL area, looking for accesses to tables **ZLIPS_##**. Try to estimate the overall impact on database usage of these statements compared to all other statements executed on database level within the time frame from database startup until now.

Solution 18: Detecting Expensive SQL Statements Using the DBA Cockpit

Task:

Analyze an expensive SQL statement using some functions of transaction ST04.



Caution: This exercise is connected with several other exercises in the ADM506 course. These exercises include the exercise on using the SQL Trace (ST05), the exercise on table statistics, and the exercise on the cost-based optimizer.

Therefore the scope of this introductory exercise might seem somewhat limited.

1.

Execute report **ZZSELECT_##** (where **##** stands for your group number, assigned to you by your instructor) several times, using transaction SE38.

Check the source code of the report you are executing using transaction SE38 and identify the name of the table accessed and the names of the fields selected.

a) Using transaction SE38, you find that report **ZZSELECT_##** accesses the table **ZLIPS_##**, using the fields *MATNR*, *WERKS* and *LGORT*.

2. Analyze the table design of the table accessed by your report, **ZZSELECT_##**, concerning existing indexes. Use transaction SE11 to answer this question.



Note: Be aware of the fact that index design and creation is covered in another lesson. It is sufficient, for this exercise, to check the existing index(es).

a) For tables **ZLIPS_##** only the primary index is available.

3. Analyze the Oracle Session Monitor and look for table accesses generated by the reports **ZZSELECT_##**. To see some entries in the Oracle Session Monitor, re-execute the report assigned to your group several times.

Continued on next page

By double-clicking the corresponding entry in the *SQL text* column, you find some information on the statement executed.

- a) Execute your report several times and (in parallel) refresh the display of the Oracle Session Monitor. Indeed it is not of much relevance to find and analyze the statement caused by your report. Finding and analyzing a statement of any **ZZSELECT_##** will do.
4. Analyze the shared SQL area, looking for accesses to tables **ZLIPS_##**. Try to estimate the overall impact on database usage of these statements compared to all other statements executed on database level within the time frame from database startup until now.
 - a) Please follow the instructions in the exercise step and the description for shared SQL Area-analysis given in the course material.

Result

You have conducted the first steps for analyzing expensive SQL statements using transaction ST04.



Lesson Summary

You should now be able to:

- Use the database performance monitor to find expensive SQL statements
- Identify SQL statements that you can tune

Lesson: Using the SQL Trace to Find Expensive SQL Statements

Lesson Overview

Using the SQL trace, you can get very detailed information on the processing of SQL statements at database level. The SQL trace can be used if you either know which function you would like to examine, or if you have a user on the system who can execute this function for you.



Lesson Objectives

After completing this lesson, you will be able to:

- Use the SQL trace to analyze the processing of SQL statements in detail on database level
- Use the Explain function

Business Example

There is a very expensive function in your productive SAP system. You would like to analyze it in some detail to find options for tuning the process at database level.

Detection of Expensive SQL Statements Using SQL Trace

The SQL Trace, transaction ST05, is a very powerful tool for analyzing database accesses. It not only offers the Explain function, but gives a list of access times for different steps executed on database level.



Note: Transaction ST05 also allows enqueue trace, RFC trace, and buffer trace.

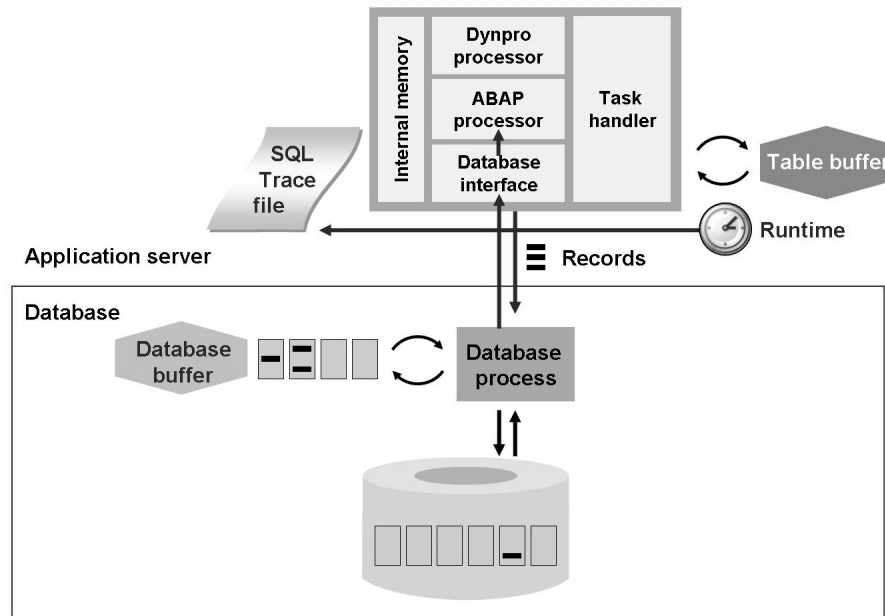


Figure 179: Using the SQL Trace for Detailed Analysis

The SQL trace works by tracing the communication between the SAP application layer and the database system. This trace is written to an SAP instance-specific file.



Hint: The size of the trace file is given by the parameter `rstr/max_filesize_MB`. The default size in SAP ECC is 16 MB. Please note that during a trace, a lot of data is written; therefore, a long-running trace might overwrite older trace data, as the trace file is written again from the start after reaching its maximum size.



Note: A search in the SAP Notes database using the term **ST05** and restricted to component area **BC*** reveals some interesting SAP Notes about the SQL trace function, for example, SAP Note 723096: *Short dump while calling Explain plan in transaction*. This note explains that with older releases (SAP Web AS 6.40 with SP2 or lower, or SAP Web AS 6.20 with SP39 or lower), an Explain fails if the SQL statement to be explained is longer than 32,000 characters. For higher SP levels of SAP Web AS, this limit has been increased to 60,000 characters.

The handling of ST05 is rather simple. To use the SQL Trace, follow these steps:



1. Execute the function you would like to trace at least once without tracing your work. This step is necessary to fill buffers.
2. Enter transaction ST05.
3. Under *Select Trace*, choose *SQL Trace*.
4. Choose *Activate Trace*.
5. Proceed, in another session, with the action you would like to trace.
6. After the action of interest has been finished, switch back to the session offering ST05 and choose *Deactivate Trace*.
7. To display the recorded trace, choose *Display Trace*.
8. The default settings in the dialog box are relevant for your last active trace. You can proceed without changing any settings, unless you would like to evaluate an older trace recording.
9. As a result, you get the **trace list**.

In the following, a very simple ABAP program, named ZSELECT, will be traced. The source code of the program is as follows.

```
REPORT  ZSELECT.

tables tadir.

select * from  tadir
where srcsystem = 'DEV'.

endselect.

write 'Done!'.
```

To understand what follows, you must understand what this little program does.

The repository information table TADIR is evaluated for entries where the given source system for repository objects is named DEV. After finishing this work, the program writes the line **Done!**. This is very simple and rather useless. However, please be aware that this simplistic program is used to explain how the database works.

More information about table TADIR can be found using transactions SE11, SE16, and DB05:

- Primary key fields: PG MID, OBJECT, and OBJ_NAME
- Two additional indexes exist : One index for the field AUTHOR and the other for the fields DEVCLASS, PG MID, and OBJECT
- Contains more than 1,800,000 entries (SAP NetWeaver 7.0)
- Has a total size of about 250 MB
- Entries are single-record buffered

Taking this information into account, it is not surprising that ZSELECT has a rather high execution time.



Duration	Obj. name	Op.	Recs	RC	Statement
4	PROGDIR	REOPEN		0	SELECT WHERE "NAME" = 'ZSELECT' AND "STATE" = 'A'
486	PROGDIR	FETCH	1	0	
8	DWAS&SYNC	REOPEN		0	SELECT WHERE "UNAME" = 'ADM&506'
555	DWAS&SYNC	FETCH	1	0	
6	DWINACTIV	REOPEN		0	SELECT WHERE "OBJECT" = 'REPS' AND "OBJ_NAME" = 'ZSELECT'
442	DWINACTIV	FETCH	0	1403	
2.854	TADIR	REOPEN		0	SELECT WHERE "SRC&S&SYSTEM" = 'DEV'
871	TADIR	FETCH	161	0	
3.145	TADIR	FETCH	161	0	
3.458	TADIR	FETCH	161	0	
8.1					
4.1					
2.1					
4.272	TADIR	FETCH	161	0	
6.946	TADIR	FETCH	161	0	
4.121	TADIR	FETCH	161	0	
2.680	TADIR	FETCH	161	0	

The execution of the statement takes almost 3 seconds.

Figure 180: Using the Trace List

How to interpret the trace list:

- In the first column, you find the time consumed by the corresponding database operation. The unit for time is microseconds: 1,000,000 ms make one second. Values higher than 100,000 ms are marked in red. There is no further meaning behind this coloring.



Hint: You can access F1 help for basic information on all fields in the trace list.

- The second column gives the name of the accessed database object.
- The third column gives the name of the database operation carried out. In all relevant cases, Explain will be carried out for operation REOPEN. The operation OPEN means that a specific database operation is carried out for the first time. Following identical operations are named REOPEN.



Note: The other possible database operations are not of central relevance for our goal. You may go to the database manufacturer's documentation to get more information on these operations.

- The next column, *Recs.*, gives the number of fetched records.
- The *RC* column gives the return code of the database system.
- The *Statement* column gives the executed SQL statement, perhaps in an abbreviated format.
- You can choose *Perform Explain for SQL statement* (F9) on a row containing the operations REOPEN or OPEN.



Observations:

- Full table scan is used
- No suitable index exists
- Old statistics data for this table

Display Execution Plan for SQL Statement

SQL Statement

```
SELECT
  *
FROM
  "TADIR"
WHERE
  "SRCSYSTEM" = :A0
```

Execution Plan

```
SELECT STATEMENT ( Estimated Costs = 1.826 , Estimated #Rows = 1.172.495 )
├── 1 TABLE ACCESS FULL TADIR
│   ( Estim. Costs = 1.826 , Estim. #Rows = 1.172.495 )
```

Table and Index Information for TADIR

Table TADIR	
Last statistics date	19.05.2004
Analyze Method	Sample 31.892 Rows
Number of rows	1.172.495
Number of blocks allocated	12.024
Number of empty blocks	7.560
Average space	922
Chain count	0
Average row length	72
Partitioned	NO

UNIQUE Index TADIR-0	
Column Name	#Distinct
PGMID	2
OBJECT	177
OBJ_NAME	862.217

NONUNIQUE Index TADIR-2	
Column Name	#Distinct
AUTHOR	2

NONUNIQUE Index TADIR-DEV	
Column Name	#Distinct
DEVCLASS	3.637
PGMID	2
OBJECT	177

☒ Index statistics

Figure 181: The Explain Function

Now that you are able to interpret the trace list, let's take a closer look at the Explain function:

- The upper part of the divided screen gives you the SQL statement and how it is executed by the database system.



Hint: Interpreting an Explain can be a rather complicated task. You will need in-depth knowledge of the database system to understand what is going on. Because some functions in database system are a manufacturer's secret, you can only guess (and wonder) at the inner workings of the system.

- The execution plan gives you details on the processing of the selected SQL statements, such as:
 - Estimated costs:** This number is the result of the access optimization by the cost-based optimizer. Please note that the actual access costs can differ from the cost-based optimizer's estimate. Oracle documentation states, "The value of this column does not have any particular unit of measurement; it is merely a weighted value used to compare costs of execution plans. The value of this column is a function of the CPU_COST

and IO_COST columns.” The columns referred to are part of the **plan table**. For more information on this topic, please see the Oracle database documentation.



Note: The estimated costs are proportional to the number of blocks necessary to read to fulfill the request (please note that “proportional” does not mean “equal.”)

- **Estimated rows:** How many rows might be read as the result of the select statement
- You are informed about the database activities executed for this statement. Examples for these activities are:
 - Table activities such as TABLE ACCESS BY INDEX ROWID
 - Index activities such as INDEX UNIQUE SCAN



Note: The execution plan is (in a simple situation) to be read from bottom to top. In more complex situations, you will find a nested information tree.



Hint: With Oracle 9i, there is a new type of access: **index skip scan**. This technique is best explained in Oracle's own words: “With Oracle 9i, a composite index [*remark by SAP: A composite index in Oracle's terminology is an index consisting of more than one field*] can be used even if the leading column(s) are not accessed by the query, via a technique called an 'index skip scan.' During a skip scan, the composite index is accessed once for each distinct value of the leading column(s). For each distinct value, the index is searched to find the query's target values. The result is a scan that skips across the index structure.”

You can find more information on the index skip scan at:
<http://otn.oracle.com/products/oracle9i/daily/apr22.html>.

The Explain screen allows for several more activities, including:

- You can **analyze** tables and indexes using the corresponding button.
- By clicking on table or index names, you can get detailed information about tables and indexes, such as the data from the last analysis and the collected statistics for this object.



Hint: By analyzing a table, the statistics are created by the database system. These statistics are used by the cost-based optimizer to determine the fastest access path for the execution of SQL statements.



Lesson Summary

You should now be able to:

- Use the SQL trace to analyze the processing of SQL statements in detail on database level
- Use the Explain function

Related Information

- Visit <http://otn.oracle.com/products/oracle9i/daily/apr22.html>.
- To better understand difficult execution plans, it is necessary to know the different processing options of the Oracle database. You can get information on these in the Oracle documentation.
- Also, it helps to have a sound understanding of programming techniques. You can learn the details of efficient ABAP programming in the SAP course *BC402 - Advanced ABAP*.

Lesson: Exclusive Lock Waits

Lesson Overview

This lesson describes the difference between database enqueues (locks) and SAP enqueues, and discusses how Oracle locks causing exclusive lock wait situations can be monitored.



Lesson Objectives

After completing this lesson, you will be able to:

- Explain when a lock wait situation occurs and how you can reduce lock waits
- Use the Lock Monitor to identify exclusive lock wait situations

Business Example

You have detected performance problems in your system, which are caused by the application design. In addition to expensive SQL statements, these problems can originate from exclusive lock wait situations. To find these exclusive lock wait situations, you can use different monitors in the SAP system.

Database Locks and SAP Enqueues

To preserve data consistency when table content is changed, the table content is locked during a change operation. This requirement is achieved by the locking concepts of the SAP system and the database system. These two locking concepts have the same purpose, but they are based on different technologies and used in different situations. Locks that are administered by the database system are called **database locks** or **database enqueues**. Locks administered by the SAP system are known as **SAP enqueues**.

Database Transactions and SAP Transactions

Every work process is connected to a specific communication partner at database level for the duration of an SAP instance's runtime. Work processes cannot exchange communication partners at runtime; this is why a work process can only make changes to the database within one database transaction.

A database transaction is, in accordance with the ACID principle (Atomic, Consistent, Isolated, Durable), a non-divisible sequence of database operations, at the beginning and end of which the dataset on the database must be consistent. The beginning and end of a database transaction are defined by a commit command (database commit) to the database system. During a database transaction (between two commit commands),

the database system itself ensures that the dataset is consistent. The database system itself takes on the task of restoring the dataset to its previous state after a transaction has terminated with an error (ROLLBACK).

Business transactions are processing units grouped to provide a specific function. These processing units execute changes to the database that are consistent and make sense in business terms. Typical examples are credit and debit postings, which only make sense together, or creating an order and reserving the relevant materials. Correspondingly, an SAP transaction is defined as a non-divisible business process that must either be executed completely or not at all. SAP transactions are implemented as sequences of logically related dialog steps that are consistent in business terms. Every user dialog step is represented by one screen image.

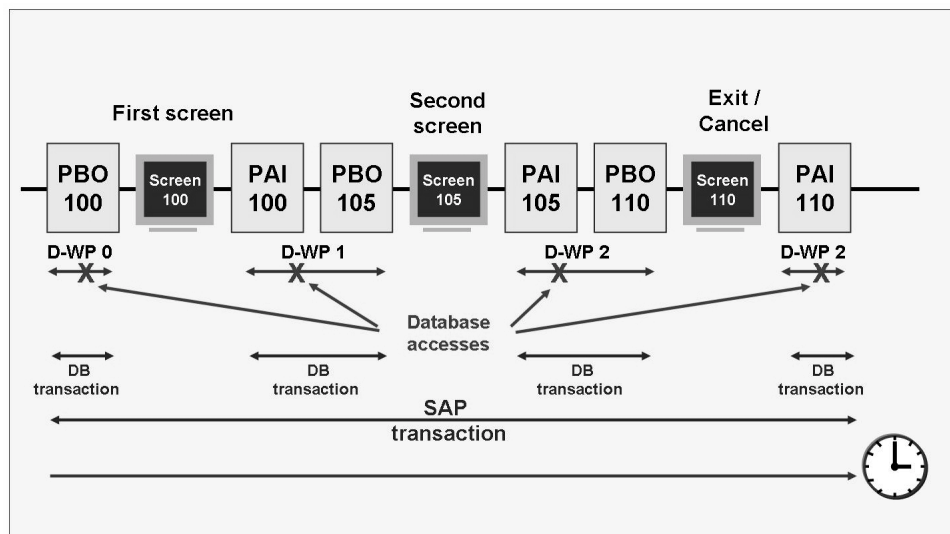


Figure 182: Database Transactions and SAP Transactions

SAP transactions are not necessarily executed within one single dialog work process. Within a transaction that changes data on the database, the user requests database changes using the displayed individual screens. Once the transaction is complete, the changes must result in a consistent database status. The individual dialog steps can be processed by different work processes (work process multiplexing), and each work process sequentially handles dialog steps for unrelated applications. Applications whose dialog steps are executed by the same work process – one after the other – cannot run within the same database transaction if they are not related to each other. Therefore, a work process must start a new database transaction for each dialog step.

The relationship between database transactions and SAP transactions is illustrated in the figure above.

Database Locks and SAP Enqueues

Business objects must not be changed simultaneously by different users if consistency is to be maintained. From the database point of view, every dialog step forms a physical and logical unit: the database transaction. The database lock administration can only coordinate this type of database transaction. From an SAP point of view, however, this is not sufficient, because SAP transactions, which are formed from a sequence of logically related work steps that are consistent in business terms, are generally made up of several dialog steps. SAP systems need to have their own lock management. This is implemented using the enqueue work process. This ensures that the platform-independence of the lock management is maintained.

Database locks are administered by the lock handler of a database instance. Typically, a row in a table is locked during a database transaction. Locks are held until the SQL statement `COMMIT` (database commit) finalizes all changes and then removes the corresponding database locks. Additionally, all database locks are removed in the case of a `ROLLBACK`.



Hint: After a transaction step, the SAP work process automatically triggers a database commit (or a database rollback). This removes all locks. This means that a database lock does not last through multiple transaction steps (through multiple input screens in the SAP system).

To lock through an SAP transaction (or SAP logical unit of work), the SAP enqueue locks a logical object. An SAP enqueue can lock rows from several different tables, or one or more complete tables if these objects form the basis of a single business document. The SAP lock concept works on the principle that SAP programs make lock entries for data records to be processed in a lock table. Lock entries can only be made if none already exist for the table entries to be locked.

The main features of database locks and SAP enqueue are outlined in the following table.



Database Locks and SAP Enqueues

	DB locks	SAP enqueues
Locked object	Rows of a table	Logical object defined in the ABAP dictionary
How object is locked	Implicitly using modifying SQL statements (such as <code>UPDATE</code> and <code>SELECT FOR UPDATE</code>)	Explicitly by the ABAP program calling an enqueue module

	DB locks	SAP enqueues
How object is removed	At the end of a database transaction or after a ROLLBACK	At the end of an SAP transaction
Maximum duration	One database transaction step	Over multiple database transaction steps
Result of lock conflicts	Wait situation, referred to as exclusive lock wait	Program-specific – for example: Maintenance of ... locked by user
How to monitor	DBA Cockpit: Lock monitor (or transaction DB01: “Exclusive Lock Waits”)	Transaction SM12: “Enqueue Monitor”

Types of Oracle Locks (Enqueues)

Oracle enqueues are always specified in the form of a two-digit ID. Oracle differentiates user enqueues and system enqueues as follows:

- User enqueues:
 - TX (transaction enqueue):

This enqueue type occurs if you want to change a data record but you cannot do this because a transaction is running in parallel (for example, this has changed the same data record because a unique or primary constraint cannot be guaranteed or because a free Interested Transaction List slot is no longer available in the block header). The TX enqueue occurs most frequently in the SAP environment. A session only ever holds one TX enqueue, even if several data records of one or several tables are changed.
 - TM (DML enqueue):

This enqueue type occurs if a complete object has to be protected against changes (for example, as part of an index rebuild or a consistency check using `VALIDATE STRUCTURE`). Whenever a TX enqueue blocks entries of a table, a TM enqueue is also set so that parallel activities such as index rebuilds or consistency checks are not possible. One TM enqueue is set for each transaction and changed object.
- System enqueues:
 - ST (space transaction enqueue):

This enqueue is held in dictionary-managed table spaces within extent allocations and releases.
 - CI (cross-instance invocation enqueue call):

This enqueue type occurs particularly if several DDL statements such as `TRUNCATE`s and `DROP`s are carried out together.

However, there are numerous other system enqueues that are generally negligible because system enqueues are only held for a very short time. The most important enqueues from a SAP point of view are TX, TM, and ST.

Monitoring Database Locks (Enqueues)

Whether the general database performance is impaired by enqueues or whether long-running transactions must wait for enqueues can be determined by a wait event analysis. The wait event analysis using transaction ST04 displays the enqueue wait event if a session is waiting for an enqueue.

To find out which sessions are currently holding enqueues or waiting for enqueues, use the V\$LOCK Oracle view. This view contains all information about sessions that hold or wait for an enqueue.



Columns of V\$Lock

SQL> describe v\$lock

Name	Null?	Type

ADDR		RAW (4)
KADDR		RAW (4)
SID		NUMBER
TYPE		VARCHAR2 (2)
ID1		NUMBER
ID2		NUMBER
LMODE		NUMBER
REQUEST		NUMBER
CTIME		NUMBER
BLOCK		NUMBER

- SID: System ID of the Oracle session that holds the enqueue or waits for it
- TYPE: Enqueue type
- ID1, ID2: Enqueue-dependent lock IDs
TM enqueue -> ID1 = Object ID of the locked object
- LMODE: Mode of the held enqueues
- REQUEST: Mode of the requested enqueues
- CTIME: Time in current mode (in seconds), that is, hold or queue time
- BLOCK: 0 -> no session is waiting for the enqueue; 1 -> at least one session is waiting for the enqueue

For sessions that are holding an enqueue, REQUEST is 0 and LMODE is higher than 0. For sessions that are waiting for an enqueue, REQUEST is greater than 0 and LMODE is 0. You can recognize those sessions in the Oracle session overview that are waiting for an enqueue (REQUEST is higher than 0) by the fact that they are waiting for the enqueue wait event.

You can identify sessions that want to access the same enqueue and are thus locked by identical entries for TYPE, ID1, and ID2.

The Lock Monitor of transaction ST04 displays the current enqueue wait situations, including the holding and waiting shadow and client process.

You can evaluate how long a session has been waiting for certain enqueues since the database was started with Oracle 9i using V\$ENQUEUE_STAT. Up to Oracle 8i, there are no enqueue statistics accumulated since the database was started.

The Enqueue Monitor sub-monitor in the DBA Cockpit helps you to monitor enqueues and reduce wait events. This monitor uses the information provided by V\$ENQUEUE_STAT. To access the Enqueue Monitor in the DBA Cockpit (transaction ST04), choose *Performance* → *Wait Event Analysis* → *Enqueue Monitor*.

The data on this screen shows which enqueue type is generating the most wait events.

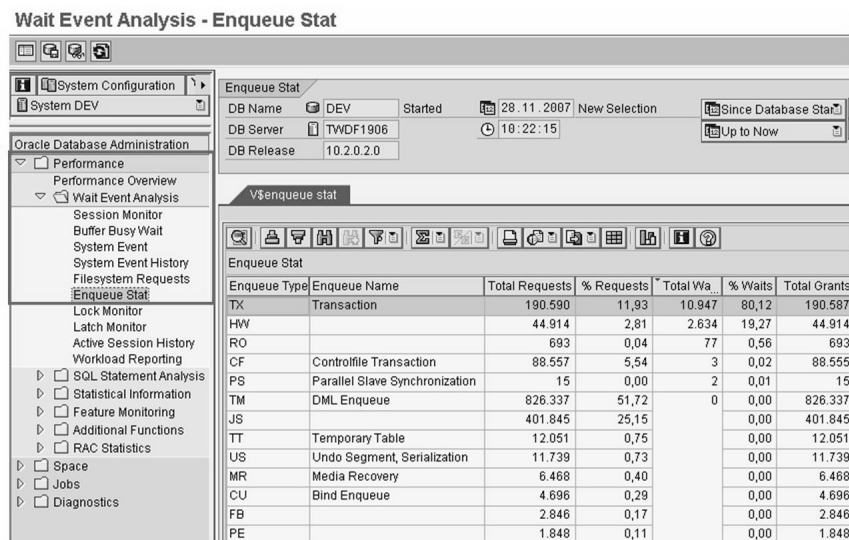


Figure 183: Enqueue Monitor

Monitoring Exclusive Lock Waits

Exclusive lock waits are wait situations that are currently being caused by database locks.

When a work process wants to lock an object that is already locked, the second process waits until the lock has been removed. This wait situation is called **exclusive lock wait**. There is no time limit placed on these locks, so if a program fails to remove a lock, the wait situation can continue indefinitely.

This could become a major problem if the program fails to release a lock. There is a danger that one work process after another will be waiting for this lock. As the number of lock waits increases, the available SAP work processes can process fewer and fewer SAP user requests. In the worst case, that is when the number of lock waits equals the number of SAP work processes, a small number of users can

cause the entire SAP system to freeze. If all work processes are waiting, there is no work process available to allow you to intervene from within the SAP system. If the program holding the problem lock can be identified, as a last alternative it can be terminated using operating system commands.

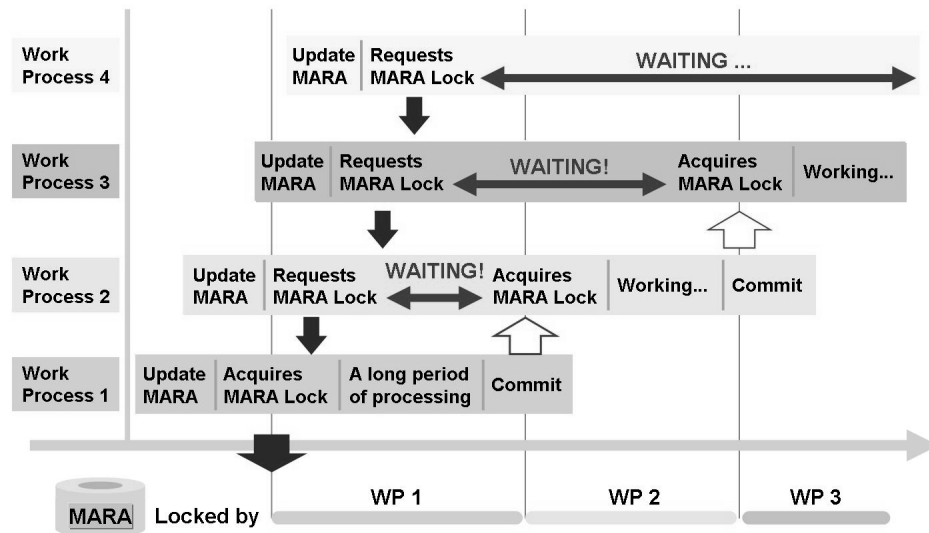


Figure 184: Exclusive Lock Wait situation

In an SAP system, exclusive lock waits can be monitored using the Oracle Lock Monitor. This monitor displays information about:



- Lock holder
- Number of lock waiters
- Primary key locked

The Oracle Lock Monitor can be called by using the DBA Cockpit (transaction ST04) and choosing *Performance* → *Wait Event Analysis* → *Lock Monitor*.

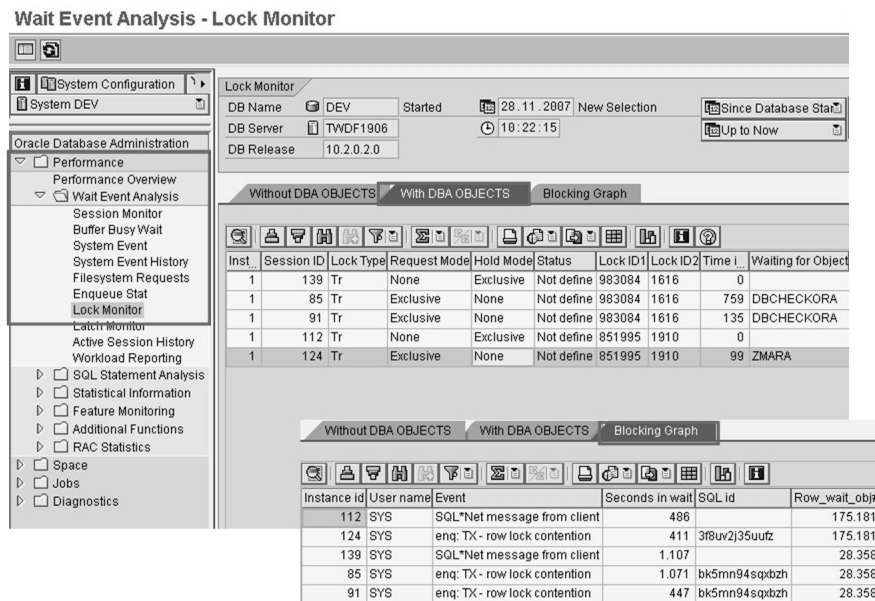


Figure 185: Monitoring of Exclusive Lock Waits

If exclusive lock waits occur, both blocking and blocked processes are displayed. For each blocking situation, a tree of waiting processes is shown. Column *PID* shows the process IDs of the host processes. Usually, this is the process ID of the SAP work process.

To display the SQL statement on the database, select a row and then choose *SQL statement*.

This sub-monitor in the DBA Cockpit helps you monitor currently active locks that are causing other requests to wait.

The following information is displayed for both the process holding the lock and the process waiting for the lock:

- Detailed lock display

You can double-click a row to view the detailed lock display, including the SQL statement.

- Related locks

From the detailed lock display, you can choose *Linked Lock* to display the related lock holders or waiters:

- For a lock holder, the lock holder itself and all lock waiters are displayed.
- For a lock waiter, the lock waiter itself and the related lock holder are displayed.

Reducing Exclusive Lock Waits

In the application programs, locks should be requested as late as possible. It is preferable for a program to read and process data from the database before setting locks or making changes in the database.

In the figure below, transaction A shows how several changes are made during the database transaction and how, as a result, database locks are held for too long. Transaction B shows a more appropriate method of programming; the transaction is programmed so that it collects the changes in an internal table and then transfers these changes to the database as a group at the end of the transaction. This reduces the lock time in the database.

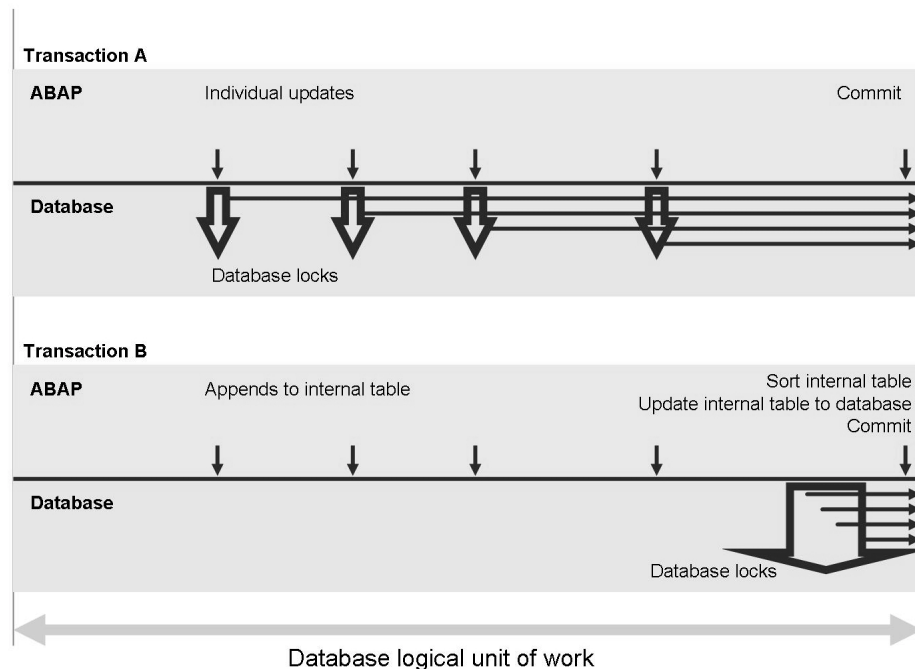


Figure 186: When Locks Should Be Set

Performance problems due to delays in releasing locks frequently occur when customers modify the programming of update modules. The separation of update modules from dialog modules aims to reduce the number of locks needed in the dialog part of a transaction, since changes to the database and the associated locks are mainly the task of the update modules. However, sometimes the update module is modified, for example, to supply a customer-developed interface with data. This modification may cause problems if the update module has already set locks and, for example, the modification generates expensive SQL statements. The locks cannot be released until the SQL statements are fully processed, and lengthy lock waits may result.

Another source of problems with locks are background programs that set locks and then run for several hours without initiating a database commit. If dialog transactions need to process the locked objects, they will be forced to wait until the background program initiates a database commit. To solve this problem, you can ensure that the background program initiates database commits more frequently, or that it runs in period with less dialog activity. Similar problems may occur when background jobs are run in parallel.

In summary, exclusive lock waits can be reduced:



- Using the SAP enqueue concept
- By a redesign of the application to reduce the locking period:
 - Increase the commit frequency of the application.
 - Single processes should not be allowed to hold a lock for a long period.
 - Objects should be locked as late as possible.
- By scheduling jobs, so that lock situations do not occur

Exercise 19: Monitoring Exclusive Lock Waits

Exercise Objectives

After completing this exercise, you will be able to:

- Monitor exclusive lock waits

Business Example

You have detected performance problems caused by exclusive lock wait situations. You want to monitor the sessions which are waiting on Oracle locks.

Task:

Start report ZZLOCK to monitor exclusive lock waits using the Lock Monitor.

1. Determine the number of transaction enqueues at the beginning and the end of this exercise and calculate the difference. Keep in mind that the enqueues of all the participants in the course are summarized.
2. Start report **ZZLOCK** in the SAP system.
3. Monitor the exclusive lock waits using the DBA Cockpit (ST04) and identify your session.
4. Monitor the Oracle wait events using the DBA Cockpit and identify the top five time-consuming wait events. Complete the table from the “Monitoring Oracle Wait Events” exercise and compare the changes in the top five wait events.

Solution 19: Monitoring Exclusive Lock Waits

Task:

Start report ZZLOCK to monitor exclusive lock waits using the Lock Monitor.

1. Determine the number of transaction enqueues at the beginning and the end of this exercise and calculate the difference. Keep in mind that the enqueues of all the participants in the course are summarized.
 - a) Call transaction ST04 and switch to the Enqueue Stat monitor. Determine the number of transaction (TX) enqueues.
 - b) Repeat this action after finishing the exercise and calculate the difference.
2. Start report **ZZLOCK** in the SAP system.
 - a) Execute report **ZZLOCK** using transaction SE38.
3. Monitor the exclusive lock waits using the DBA Cockpit (ST04) and identify your session.
 - a) Call transaction *ST04* and switch to the Lock Monitor.

The Lock Monitor displays the process ID of the SAP work processes waiting for the lock.

You can identify your session by using the process monitor (SM50) to find the process ID of your work process.
4. Monitor the Oracle wait events using the DBA Cockpit and identify the top five time-consuming wait events. Complete the table from the “Monitoring Oracle Wait Events” exercise and compare the changes in the top five wait events.
 - a) Call transaction DBACOCKPIT or ST04 and switch to the System Events monitor (choose *Performance* → *Wait Event Analysis* → *System Events*).

The Wait event details sub-monitor displays the non-idle wait events. The list is sorted by wait time in descending order.



Lesson Summary

You should now be able to:

- Explain when a lock wait situation occurs and how you can reduce lock waits
- Use the Lock Monitor to identify exclusive lock wait situations



Unit Summary

You should now be able to:

- Explain why even a few expensive SQL statements may reduce performance for the entire SAP system
- Use the work process overview to find potentially expensive SQL statements currently executed in your SAP system
- Use workload analysis to identify transactions in SAP systems that bring significant load to the database
- Use statistical records to identify transactions in SAP systems that bring significant load to the database
- Use the database performance monitor to find expensive SQL statements
- Identify SQL statements that you can tune
- Use the SQL trace to analyze the processing of SQL statements in detail on database level
- Use the Explain function
- Explain when a lock wait situation occurs and how you can reduce lock waits
- Use the Lock Monitor to identify exclusive lock wait situations

Unit 9

Index management and optimization

Unit Overview

This unit describes in the first lesson how an index is structured and pointed out how data is accessed using an index. In second lesson lesson explains the rules for creating an index and displays how you create an index in the SAP system.



Unit Objectives

After completing this unit, you will be able to:

- Describe what an index is
- Explain how an index is structured
- Discuss how data is accessed using an index
- Create an index
- Figure out what kind index should be created
- Check for missing indexes

Unit Contents

Lesson: Index Utilization	534
Lesson: Creating an Index.....	551
Exercise 20: Creating an Index.....	565

Lesson: Index Utilization

Lesson Overview

This lesson describes how an index is structured and how data is accessed using an index.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe what an index is
- Explain how an index is structured
- Discuss how data is accessed using an index

Business Example

You have detected performance problems in your system, which are caused by missing or inaccurate indexes. You need to collect information about what an index is and how data is accessed using an index.

Fundamentals of Database Organization

The data inserted in a database table is generally not sorted. New data is either added -- unsorted -- to the end of the table, or entered in empty spaces where records were deleted from the table. This is shown in the following database table, which lists the members of a sports club. Focus is put on the columns *M-NR* (membership number), *LNAME* (last name), and *FNAME* (first name).



Database Table: MEMBERS

M-NR	LNAME	FNAME
111	Davis	Ricky
333	Nowitzki	Dirk
444	Davis	Dale
123	Jordan	Michael
222	Davis	Antonio

The content of a table is physically stored in multiple blocks that are distributed over one or more data files. One way for the database to access data from an unsorted table is to use a time-consuming full table scan, record for record. The effort for the database increases linear with the size of the table.



Datafile 7							
Block 47		Block 48		Block 49		Block 50	
1	111 Davis Ricky	1	123 Jordan Michael	1	531 ...	1	716 ...
2	236 ...	2	222 Davis Antonio	2	577 ...	2	555 ...
3	187 ...	3	284 ...	3	432 ...	3	644 ...
4	438 ...	4	184 ...	4	444 Davis Dale	4	333 Nowitzki Dirk

Figure 187: Data Structure

Data searches in the database can be accelerated using indexes. Just like an index in a book, which helps you find information faster, an index placed on a table helps you retrieve data faster. An index is a database feature (a list of keys or keywords) that allows searching and locating data quickly within a table. Indexes are created for frequently searched attributes (table columns) in order to optimize the database performance.

An index is merely a fast access path to the data. It affects only the speed of execution. Given a data value that has been indexed, the index points directly to the location of the rows containing that value. The absence or presence of an index does not require a change in the wording of any SQL statement.

Indexes are logically and physically independent of the data in the associated table. You can create or drop an index at any time without affecting the base tables or other indexes. If you drop an index, all applications continue to work. However, access to previously indexed data can be slower. Indexes, as independent structures, require storage space.

Oracle Index Structure

Oracle provides several indexing schemes, which provide complementary performance functionality:



- B* tree indexes
- Bitmap indexes

B* tree indexes

A normal B* tree index is a tree made up of blocks with up to five levels. The highest level consists of only one block, which is called the root block and which is the basis for every index access. The lowest level consists of a large number of blocks that are called leaf blocks; leaf blocks contain the actual data from left to right in a sorted order (values of the indexed columns and row ID for access to the relevant table entry). The entries are called leaf rows.

Depending on the size of the index, additional index levels with branch blocks may exist between the root blocks and leaf blocks; these contain control data for a fast search of the relevant data in the index.

If an index only consists of one level, the root block is also a leaf block and contains the actual data. If an index consists of several levels, the root block can be compared with the branch blocks and contains information about navigating in the index.

In general, an index mainly consists of leaf blocks. If there is a very large number of index entries or if a COALESCE has been executed for a fragmented index, there may be a large number of branch blocks.

An index consists of one or more fields of a table. Every row in the table is identified by the row ID, which is also the link between the table and the index. The row ID consists of the Oracle data file number, the number of the Oracle data block within the data file, and the row number inside the data block. The rows are sorted in the index by the indexed columns to allow for a quick access. In the table the rows are not sorted.

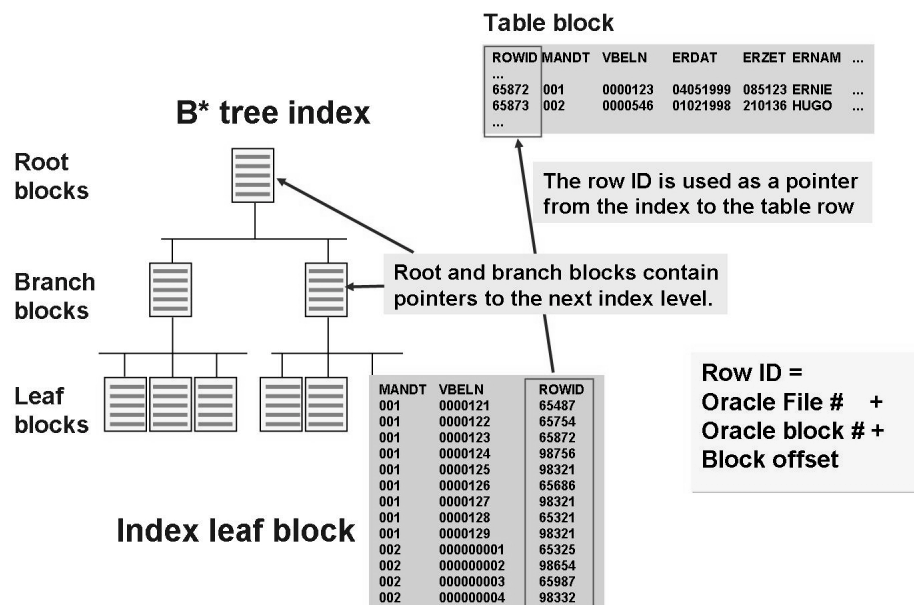


Figure 188: B* Tree Index Structure

An index should not have too many fields. If an index has few very selective fields, its reusability increases and the possibility of a bad optimizer decision decreases. The B* tree of an Oracle index in the SAP environment normally has up to four levels. When the database searches for a row in an index, the root block is read first, followed by blocks on the subsequent branch levels and the leaf level. The root and branch blocks contain pointers to the following level, sometimes called separators. These separators contain an abbreviated index key and allow for efficient access to the necessary index leaf block. The leaf blocks contain the full index rows and the row ID for the records. When you search for a row in the index, no more than four blocks usually have to be read, that is, one block for every index level.

Bitmap indexes

A bitmap index is also physically organized like a B* tree, but a bitmap for each key value is used instead of a list of row IDs.

Each bit in the bitmap corresponds to a possible row ID. If the bit is set, the row with the corresponding row ID contains the key value. A mapping function converts the bit position to an actual row ID, so the bitmap index provides the same functionality as a regular index even though it uses a different internal representation. If the number of different key values is small, then bitmap indexes are very space efficient.

The following figure is an example of a bitmap index. The table entries are stored in two different blocks of Data file 9. The leaf blocks of the index contain the key value (in this example the key value includes only the *GENDER* field, with the values male and female), the start-row ID and the stop-row ID, and the bitmap segment.



Table:

NAME	FNAME	GENDER
Christensen	Hayden	m
Portman	Natalie	f
Wood	Elijah	m
Mortensen	Viggo	m
Tyler	Liv	f
Otto	Miranda	f
Fisher	Carrie	f

Bitmap Index

Key	Start Row ID	Stop Row ID	Bitmap
Female	9 15 02	9 16 04	0100111
Male	9 15 02	9 16 04	1011000

Data File 9

Block 15	1	
	2	Christensen Hayden m
	3	Portman Natalie f
	4	Wood Elijah m
Block 16	1	Mortensen Viggo m
	2	Tyler Liv f
	3	Otto Miranda f
	4	Fisher Carrie f

Figure 189: Bitmap Indexes

The advantage of using bitmap indexes is greatest for low-cardinality columns, that is, columns in which the number of distinct values is small compared to the number of rows in the table.

AND and OR conditions in the WHERE clause of a query can be quickly resolved by performing the corresponding Boolean operations directly on the bitmaps before converting the resulting bitmap to row IDs. If the resulting number of rows is small, the query can be answered very quickly without resorting to a full table scan.

Bitmap indexing benefits data warehousing applications that have large amounts of data and *ad hoc* queries, but a low level of concurrent transactions. Bitmap indexes are not suitable for OLTP applications with large numbers of concurrent transactions modifying the data. These indexes are primarily intended for decision support in data warehousing applications where users typically query the data rather than update it.

The indexes normally set up by the SAP system are B* tree indexes. Bitmap indexes are particularly significant in the SAP NetWeaver BI system (for example, on the characteristic columns of InfoCubes).

Types of Indexes

Different types of indexes are:



- Primary index
- Secondary index
- Unique secondary index

All tables of the SAP system have a primary index, which consists of the key fields that the developer must define when creating the table. An index can be defined as unique if a maximum of one record exists for any combination of fields in the index. The primary index is always unique because the key fields must identify each table record uniquely.

In the MEMBERS table example, the primary index might be sorted numerically by the membership number. Because this index is structured, the database does not need to read the entire list sequentially; instead, the database can expedite the search by first searching this index for the membership number and then using the corresponding row ID.

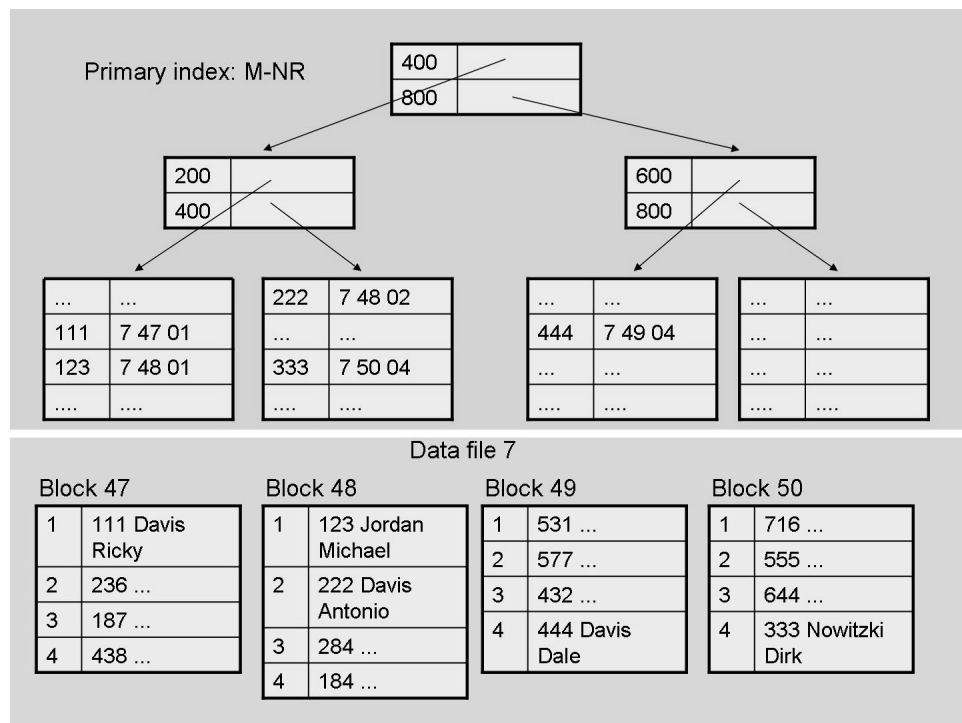


Figure 190: Primary Index

For queries where the primary index cannot be used to determine the results, for example, because none of the primary index fields were specified in the WHERE clause, the RDBMS scans through the entire table. To avoid this, you can define secondary indexes to significantly reduce the number of data records searched. Secondary indexes can also be unique indexes.

In our example, the primary index helps you only if you already know the membership number. If you had only the last name and wanted to look up the corresponding data for this member, the primary index would not be useful because it does not contain the field *LNAME*. Consequently, you would have to sequentially read the entire table of members. To simplify search queries of this kind, you could define a secondary index containing the field *LNAME* and the corresponding row IDs. This secondary index is not unique because there are multiple entries for the *LNAME* field.

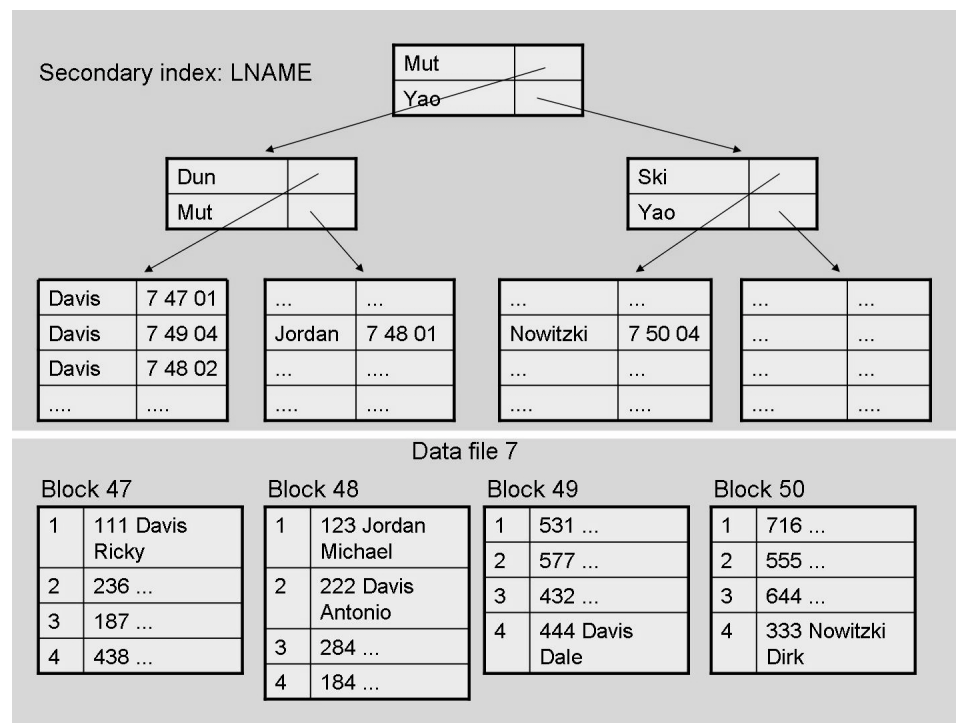


Figure 191: Secondary Index

You cannot define a single index to match all field combinations used in queries. However, to keep unnecessary data searches to a minimum, ensure that the database can always use an appropriate index.

Possible Data Access Paths

Index Unique Scan

An index unique scan always returns a maximum of one data record. If all key fields of a unique index are specified in the WHERE clause of the SQL statement, the row ID is found with an index unique scan. The row ID is then used to access the row requested in the table. Approximately four blocks are accessed to find the row. In this example, table ZVBAK has a unique index with the fields *MANDT* and *VBELN*, which are the client and document number. This index is used to find the row where *MANDT* = 001 and *VBELN* = 0000123. Three blocks have to be read on the index and another block on the table.



```
SELECT * FROM ZVBAK WHERE MANDT = '001' AND VBELN = '0000123'
```

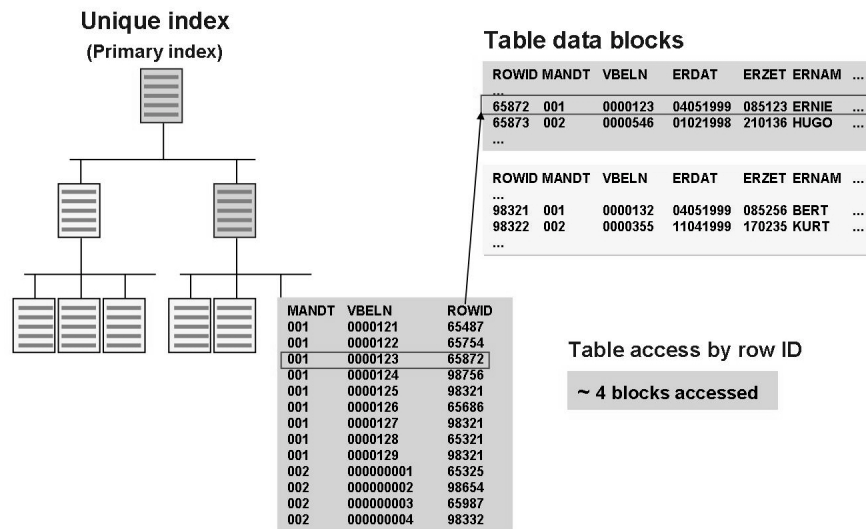


Figure 192: Index Unique Scan

Index Range Scan

In an index range scan, the system scans a subset of the leaf blocks and can return as many data records as required. If the key fields of a unique index are not specified with “=” in the WHERE clause of the SQL statement, or if a non-unique index is used, the result set is found with an index range scan. If an index is defined as a secondary (non-unique) index, the result set is also found with an index range scan.

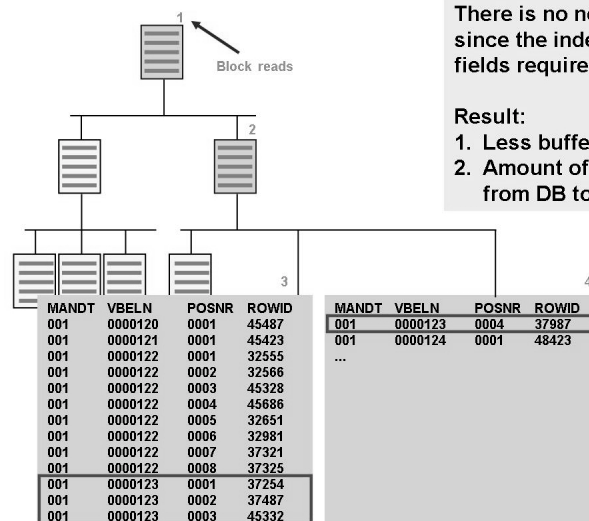
In this example, table ZVBAP has a unique index on fields *MANDT*, *VBELN*, and *POSNR*. However, in the query, only *MANDT* = '001' and *VBELN* = '0000123' are specified. For this criteria, four records exist in the table with different values for the *POSNR* field. In the index, these records are distributed across two index leaf blocks and two table blocks.

The order of fields in an index is important for the efficiency of the index scan. In the example below, a query is sent to the database which requests all the records of table ZVBAP where *MANDT* = '001' and *VBELN* = '0000123'. The data in the index is sorted by the criteria *MANDT*, *VBELN*, *POSNR*. That means the database only has to scan the index for entries where *MANDT* = '001' until *VBELN* > '0000123'. In other words, the searched string on the index is 0010000123%.

If the index already provides all data fields required by the select clause, the data blocks do not need to be accessed anymore. The result set in this example is found with an index range scan.



```
SELECT VBELN, POSNR FROM ZVBAP WHERE MANDT='001' AND VBELN='0000123'
```



There is no need to access the data blocks, since the index already provides all data fields required by the SELECT clause.

Result:

1. Less buffer gets
2. Amount of transferred data (network load) from DB to ABAP is reduced

Searched index string:
0010000123%

Figure 193: Index Range Scan

If the index does not provide the required data fields, the row IDs are then used to access the rows requested in the table. In the example below, three row IDs found on the first index leaf block and two table blocks have to be read. For the row found on the next index leaf block, the table block read for the previous index block has to be read again. In total, seven blocks have to be read: the root and branch block of the

index, two leaf blocks of the index, and two blocks on the table, one of which is read twice (once for position numbers 1 and 2 and the second time for position number 4, which is contained in a different index block, but the same table block).

The block that is read first on the index leaf level is found by the separators in the index root and branch blocks. This index is efficient.

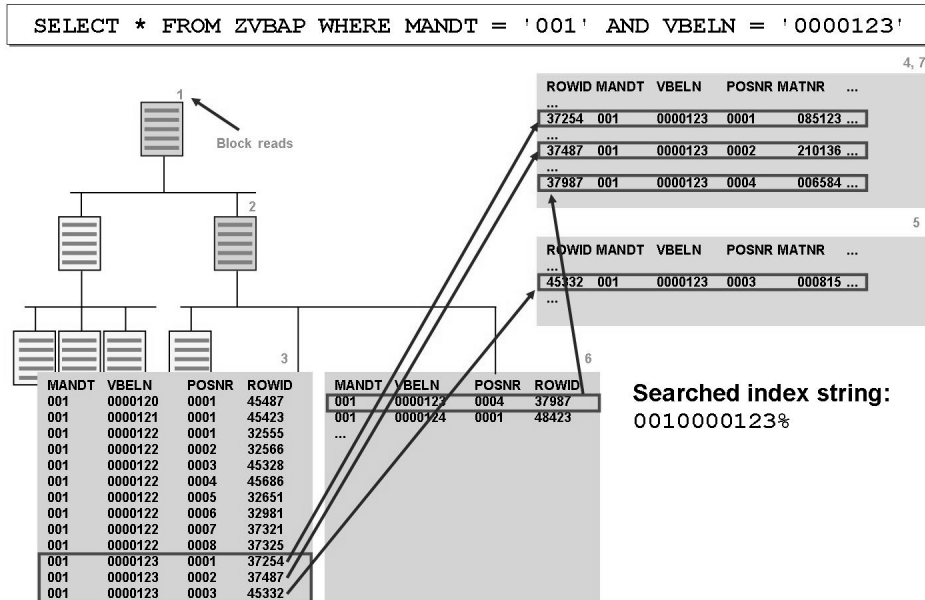


Figure 194: Index Range Scan: Table Access by Index Row ID

In the next example, the index has a different order, but the SQL statement is the same as in the previous figure. The fields specified in the WHERE clause are not the first fields used in the index. The position number is not specified in the query, but is contained in the index in the second position between the client and the document number, which are both specified. The index is now sorted by the criteria: MANDT; POSNR; VBELN. All the index blocks for MANDT='001' have to be read because it is unknown which POSNR records exist in the table with VBELN='0000123'. In other words, the searched index string is 001____0000123.

This index scan is inefficient. Many index blocks are read that do not contain data requested by the query. The fields that are specified sequentially (starting with the first field) are used to limit the access to index blocks. Any other fields that are specified for the index can be used to limit the access to data blocks. This is called a **filter**.

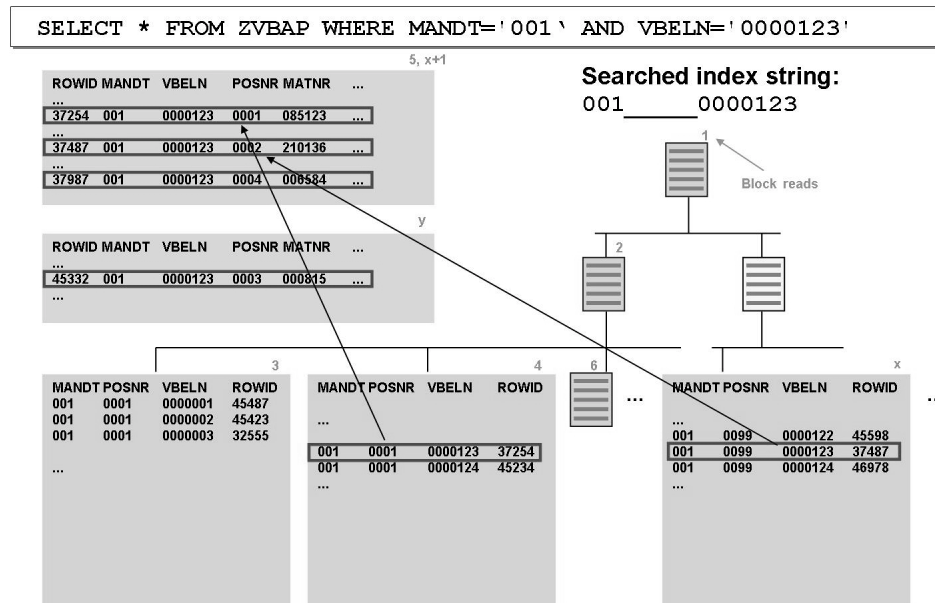


Figure 195: Order of Fields in the Index (Index Range Scan; Table Access by Index Row ID)

Unselective Index Range Scan

During an unselective index range scan, the number of gets per execution can be much higher than the number of table blocks and the number of index blocks. The same table block can be read multiple times if its rows are requested from multiple index blocks. The maximum number of blocks that can be read is determined by the number of table rows, because every row contained in each index block can be read from a different table block. Each time an index or data block is accessed, the entire 8 KB of data is read. If a large fraction of the index is read, a full table scan would be more efficient.

All data and index blocks that are read during an index range scan are placed in the middle of the least recently used (LRU) list and are therefore held in the buffer. This can cause the database to swap a lot of other blocks out of the buffer. Therefore, this type of query can cause many disk accesses for other statements. The reasons for choosing an unselective index could be outdated optimizer statistics or an optimizer bug, or parameter settings. In this example, the field *MANDT*, which is very unselective, is specified in the WHERE clause of the SQL statement and is part of the index used. The selective field *MATNR* however, is not part of the index. Therefore, all the rows where *MANDT* = '001' have to be read on the table. Most of them do not have the requested *MATNR* and are not passed back to the application.



```
SELECT * FROM ZVBAP WHERE MANDT = '001' and MATNR = '000815'
```

Index

Table ZVBAP

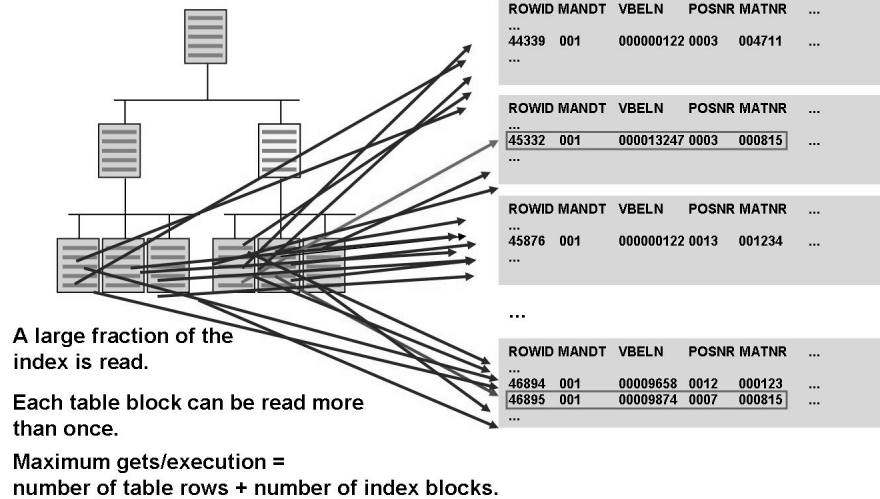


Figure 196: Unselective Index Range Scan

Index Skip Scan

An index skip scan may be useful if indexed columns are not specified or if they are only specified with a range condition. It is technically comparable to several index range scans.

Index skip scans are a new performance feature in Oracle 9i, and can provide significant benefits to any database application that uses composite indexes.

A composite index (also called a concatenated index) is an index that contains multiple columns. In releases prior to Oracle 9i, a composite index would only be used for a given query if either the leading column (or columns) of the index was included in the WHERE clause of the query, or if (less often) the entire index was scanned.

With Oracle 9i, a composite index can be used even if the leading column(s) are not accessed by the query, via a technique called an index skip scan. During a skip scan, the composite index is accessed once for each distinct value of the leading column(s). For each distinct value, the index is searched to find the query's target values.

Index skip scans are basically a series of index range scans on logical partitions of a single index. Skip scans involve range scanning N logical indexes, where N is the number of distinct values in the leading column of the index. Range scanning is best done when you are retrieving a small subset of rows.

The result is a scan that skips across the index structure. The index is sorted by the criteria: MANDT; LGORT; VBELN. If the field *LGORT* has only two distinct values, an index skip scan may be preferable to a full table scan.



```
SELECT * FROM ZVBAP WHERE MANDT='001' AND VBELN='0000123'
```

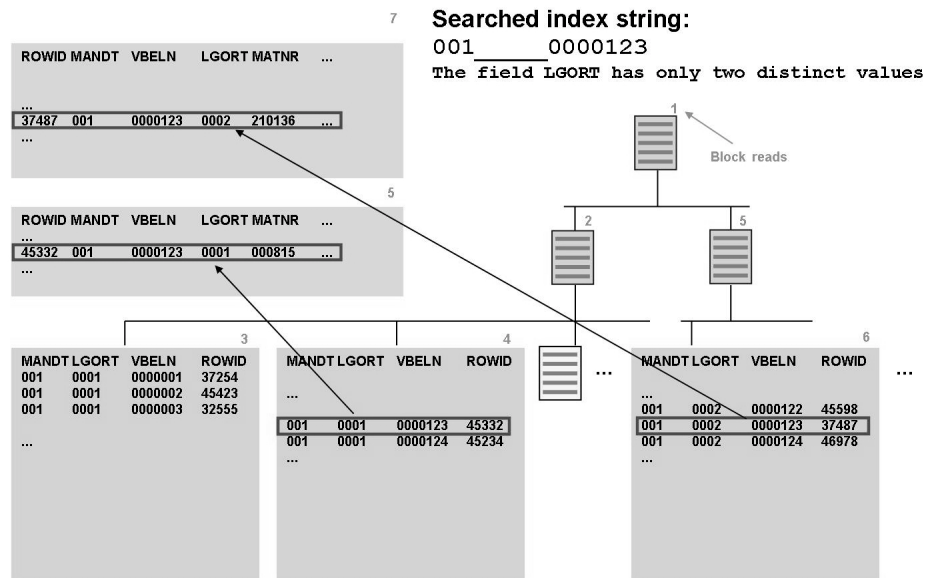


Figure 197: Index Skip Scan

Index skip scans provide two main benefits:

- Index skip scans can improve the performance of certain queries, since queries that previously required table scans may now be able to take advantage of an existing index.
- Index skip scans may allow an application to meet its performance goals with fewer indexes. Fewer indexes will require less storage space and may improve the performance of DML and maintenance operations.

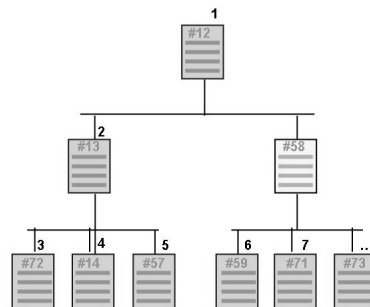
Index Full Scan

There are some SQL queries that can be resolved by reading the index without touching the table data.

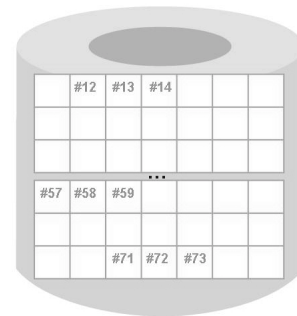
Use an index full scan all the leaf blocks of the index are scanned in the logical tree structure. That means that the system walks down the branch blocks to the first (lowest key) leaf block, and then walks the index in key order, typically reading one leaf block at a time, and using the leaf pointers to get from leaf block to leaf block. This operation can be set to work in descending order through the index.



Index: Logical Structure



Database: Physical Structure



Index full scan: All the leaf blocks of the index are scanned in their logical tree structure.

Figure 198: Index Full Scan

As a prerequisite for an index full scan, all of the columns required must be specified in the index. That is, all columns in the SELECT and WHERE clauses must exist in the index.

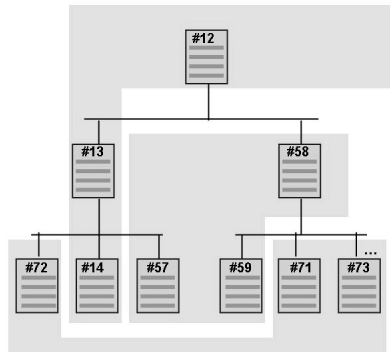
Index Fast Full Scan

In an index fast full scan, all the blocks of the index are scanned in their physical sequence. An index fast full scan is an enhancement of the index full scan. A fast full scan is faster than a normal full index scan in that it can use multi-block I/O and can be parallelized just like a table scan. The fast full index scan is an alternative to a full table scan when there is an index that contains all the keys that are needed for the query.

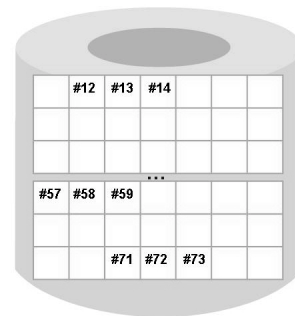
Using an index fast full scan, all the blocks of the index are scanned in their physical sequence. That means the system goes to the first block of the segment and does multi-block reads through the segment, picking up branch and leaf blocks, discarding the branches, and using the data in the leaf blocks as if they were skinny tables. In the example below only two disk I/O accesses are necessary.



Index: Logical Structure



Database: Physical Structure



Index fast full scan: All the blocks of the index are scanned in their physical sequence.

Figure 199: Index Fast Full Scan

The fast full index scan is almost always used for count(*) operations.

Full Table Scan

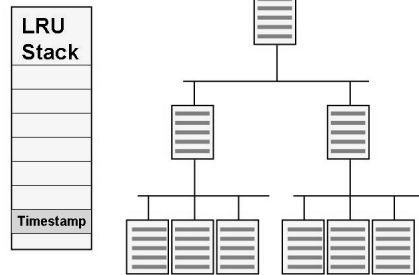
If a full table scan is used, all data blocks of the table are read sequentially but no index blocks are read. For a full table scan, the data blocks read from disk are placed at the end of the LRU list. Therefore, only a small amount of memory is used in the data buffer for full table scans. This access path is efficient if a large fraction of the table rows are requested by the query. However, if only few table rows are requested, then a full table scan is very inefficient.

Each statement enters the LRU list with a time stamp. The time stamp is refreshed every time the statement is called again. If the free space in the cache is exhausted, the data and index block of statements with the oldest time stamps are displaced from the cache. Full table scan statements are initially stored at the end of the LRU list. Thus, blocks from full table scans are displaced earlier than other statements.



```
SELECT * FROM ZVBAP WHERE MATNR = '000815'
```

Index



No index is used during a full table scan.

Each data block is read once during a full table scan.

Table ZVBAP

ROWID	MANDT	VBELN	POSNR	MATNR	...
...	44339	001	000000122	0003	004711
...					
...	45332	001	000013247	0003	000815
...					
...	45876	001	000000122	0013	001234
...					
...					
...	46894	002	00009658	0012	000123
...	46895	002	00009874	0007	000815
...					

Maximum gets/execution = number of table blocks.

Figure 200: Full Table Scan



Lesson Summary

You should now be able to:

- Describe what an index is
- Explain how an index is structured
- Discuss how data is accessed using an index

Lesson: Creating an Index

Lesson Overview

In this lesson, you will learn the rules for creating an index and how to create an index in the SAP system.



Lesson Objectives

After completing this lesson, you will be able to:

- Create an index
- Figure out what kind index should be created
- Check for missing indexes

Business Example

You have detected performance problems in your system, which are caused by missing or inaccurate indexes, and you have collected more information about what an index is and how data is accessed using an index. Now you need to know how to create an index and how to find out which index should be created.

Rules for Creating or Changing Indexes

Creating or changing a secondary index changes the SAP system and can improve or worsen the performance of SQL statements. Therefore, changes to indexes should be performed only by experienced developers or consultants.

Preliminary Checks

Before creating or changing an index, check whether the SQL statement for which you want to create a new index originates from a standard SAP program or a customer-developed program.

- If the SQL statement originates from a standard SAP program, check the relevant SAP Notes in the SAP Service Marketplace that describe ways of optimizing performance for the SQL statement. If there are no relevant SAP Notes, enter your proposal concerning the creation of an appropriate index in a problem message in the SAP Service Marketplace.
- When optimizing customer-developed SQL statements, try to avoid creating secondary indexes on SAP transaction data tables. As a rule, transaction data tables grow linearly over time and cause a corresponding growth in the size of the related secondary indexes. Over time, searching in a secondary index will result in an SQL statement that runs more and more slowly. For transaction data, therefore, SAP uses special search techniques such as matched tables and SAP business index tables like the delivery due index.

If the SQL statement originates from a customer-developed program, rather than create a new index, you may be able to:

- Rewrite the ABAP program in such a way that an available index can be used.
- Adapt an available index.

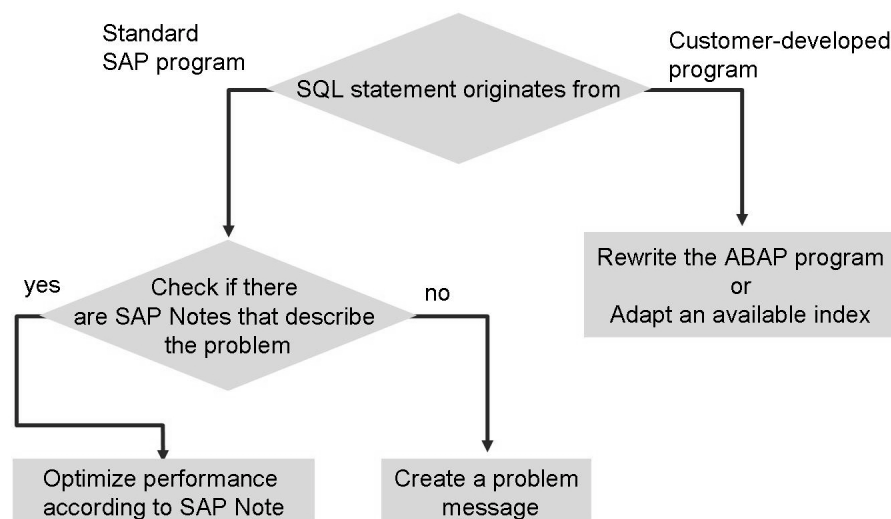


Figure 201: Preliminary Checks

Never create a secondary index on SAP Basis tables without explicit recommendation from SAP. Examples of these tables include table NAST and tables beginning D010, D020, and DD.

Rules for Creating Indexes

The following rules are the basic rules of secondary index design. For primary indexes, there are other considerations related to the principles of table construction in addition to these rules.

- **Include only selective fields.**

An index is useful only if each corresponding SQL statement selects only a small part of a table.

Examples of selective fields normally include document numbers, material numbers, and customer numbers. Examples of nonselective fields usually include SAP clients, company codes, or plant IDs.

- **Include as few fields as possible.**

As a rule of thumb, an index should contain no more than four fields. If too many fields are used in the index, it has the following effects:

- Change operations to tables take longer because the index must also be changed accordingly.
- More storage space is used in the database. The larger volume of data in the index reduces the chance that the optimizer will regard it as economical to use the index.
- The parsing time for an SQL statement increases significantly, especially if the statement accesses multiple tables with numerous indexes and the tables must be linked with a join operation.

- **Position selective fields to the left in an index.**

From a performance point of view, it does not matter whether selective or non-selective fields (such as *MANDT*) are located at the beginning of the index. It is much more important that fields that are not specified with "=" in WHERE conditions, and that cover a large value range, are located as far at the bottom of the index as possible. In addition, the index should not contain any gaps that are not specified in the WHERE condition.

To speed up accesses through an index based on several fields, the most selective fields should be positioned furthest toward the left in the index.

- **Indexes should be disjunct.**

Avoid creating nondisjunct indexes – that is, two or more indexes with largely the same fields.

- **Create as few indexes per table as possible.**

As a rule of thumb, you should not create more than five indexes for each table. One exception is for a table that is used mainly for reading, such as a table containing master data. Having too many indexes causes similar problems to those that occur if an index has too many fields. There is also an increased risk that the optimizer will choose the wrong index.

In systems using the rule-based optimizer, there is a high risk that an index could be used unintentionally.

- **Do not change an index created by SAP.**

Never change an index created by SAP, unless you are requested to do so by SAP.

Keep in mind that every rule has exceptions:

- To make sure that the optimizer uses an index, it is sometimes necessary to use nonselective fields in the index in a way that contradicts the first three rules. Examples of such fields typically include the fields for client ID (field *MANDT*) and company code (field *BUKRS*).
- Sometimes the optimal index combination can be found only by trial and error. Generally, experimenting with indexes is considered safe, as long as you keep the following points in mind:
 - Never change indexes for tables larger than 10 MB during company business hours. Creating or changing an index can take from several minutes to several hours and blocks the entire table. This causes serious performance problems during production operation.
 - After creating or changing an index and creating new statistics for the table, always check whether the optimizer program uses this index in the manner you intended. Ensure that the new index does not result in poor optimizer choices for other SQL statements.

Selectivity Analysis

Before creating an index, you must know how selective the index will be. To display the distribution of records for possible field and value combinations of the index, use the “Analysis of table with respect to indexed fields” (transaction DB05).

Use transaction DB05 for a rough overview of what and how often value combinations of one or several columns appear in the table. DB05 does not state the column combination appears how often. In addition, the frequency is only broken down by intervals (1-10, 11-100, and so on). The result of a DB05 analysis is therefore of limited use.

DB05 is useful for columns with a limited number of value combinations to determine each of these combinations, including the exact number of occurrences.

You can use SQL*Plus to determine the index selectivity. The following SQL statement will return the combinations that occur more than a hundred times:

```
select
Field1, Field2, Field3, count(*)
from Table
group by Field1, Field2, Field3
having count (*) > 100 order by count (*);
```

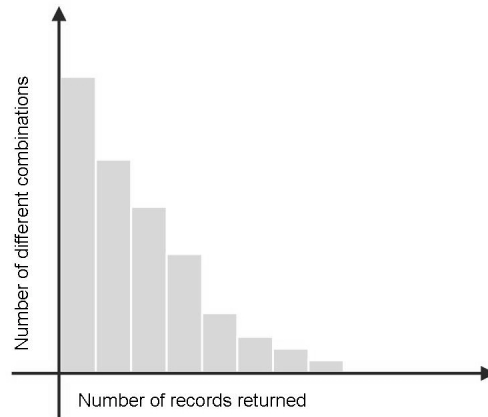


Hint: Selectivity analysis using either transaction DB05 or SQL*Plus is expensive.

An index range scan is cheap if only a few records are found using the index. A histogram can be calculated to ensure that there are no combinations of values that would return many columns. The table contents are scanned for the different possible combinations of the indexed fields. For each combination, the number of records that are returned is recorded. In the histogram, the number of different combinations that return a certain number of records is recorded.



How many records will be returned by the index scan?



Index: Field1, Field2, Field3

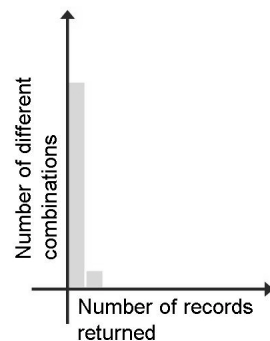
```
SELECT FROM TABLE
WHERE FIELD1 = 'A'
AND   FIELD2 = 'B'
AND   FIELD3 = 'C'
```

Figure 202: Histogram of Combinations

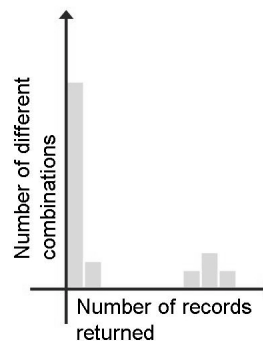
The following figure shows three different histograms of combinations according to the variable selectivity of an index. These are “A selective index”, “An index that is selective for most combinations of indexed fields”, and “An index that is not selective for most combinations of indexed fields.”



A selective Index



An index that is selective for most combinations of indexed fields



An index that is not selective for most combinations of indexed fields

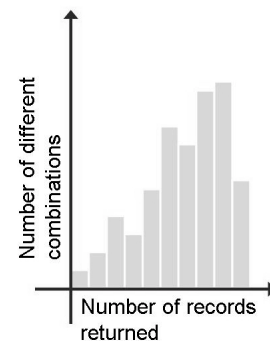


Figure 203: Histograms of Combinations

The histogram on the left-hand side shows an index that is selective for all of the possible field combinations.

If the index is selective for most of the combinations of indexed fields, a histogram similar to the one displayed in the middle will result. Most of the combinations return only a few records, and some combinations of fields exist that would return a lot of records. In this case you must check if a query could be issued by the application with the combinations of values for which many records would be returned. This would cause an expensive unselective index range scan. Only the developers of the table and the programs that are selecting data from the table can decide if this should happen.

For example, assume that a spool table exists where all the print requests are stored. All requests have a status flag, `STATUS`, that can be set to either *printed* or *not printed*. The majority of the spool requests are usually in the status *printed* and only a few are *not printed*. The histogram for an index with the fields `PRINTER` and `STATUS` will show some combinations for which only a few records would be returned by the index, that are the not yet printed print requests to a certain printer. Other combinations would return many records, including all the print requests that are already printed on a printer. Only if it can be assured by the application and table logic that the already printed print requests are not selected from the spool table using the `STATUS` flag, the index could be created to accelerate the search of the not-yet-printed print requests.

An index that usually returns many records will show a histogram similar to the histogram displayed on the right-hand side. Such an index should not be created.

Transaction DB05 displays the selectivity of a field combination. Use this information to determine if an index should be created. The number of distinct values shows you how many different combinations of fields exist. This number should be close to the number of table rows for the combination of all of the key fields of the index.

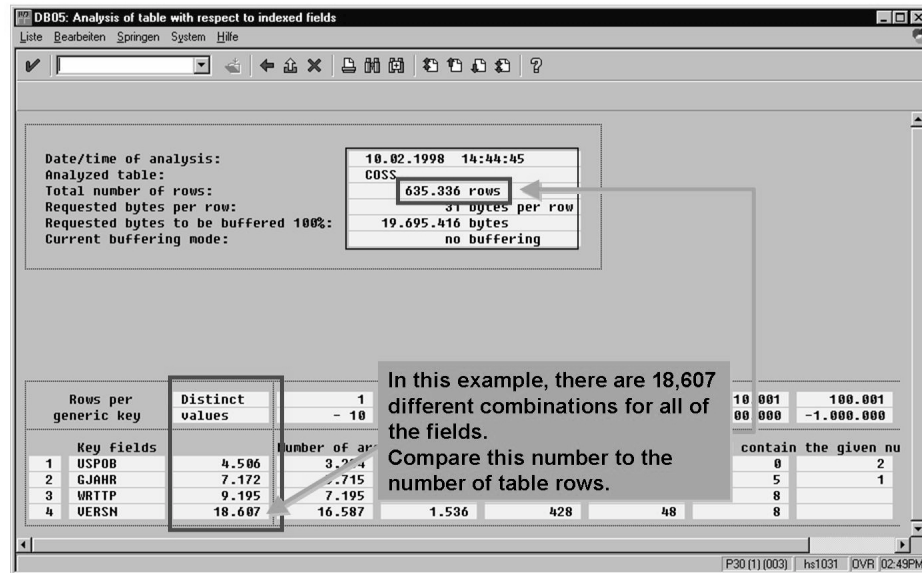


Figure 204: Example of Selectivity Analysis

In the example above, there are:

- 4,506 different values for field USPOB
- 7,172 possible combinations of USPOB and GJAHR
- 9,195 possible combinations of USPOB, GJAHR, and WRTTP
- 18,607 possible combinations of all the fields
- 635,336 table rows

When you compare the possible combination of all the fields to the number of table rows, you can determine that this index would return 30 rows on average.

The following aspects should be considered when interpreting the results of the “Analysis of Table with Respect of Indexed Fields” (transaction DB05).

If the number of distinct values does not increase for a field, the field should not be included in the index (see figure below).

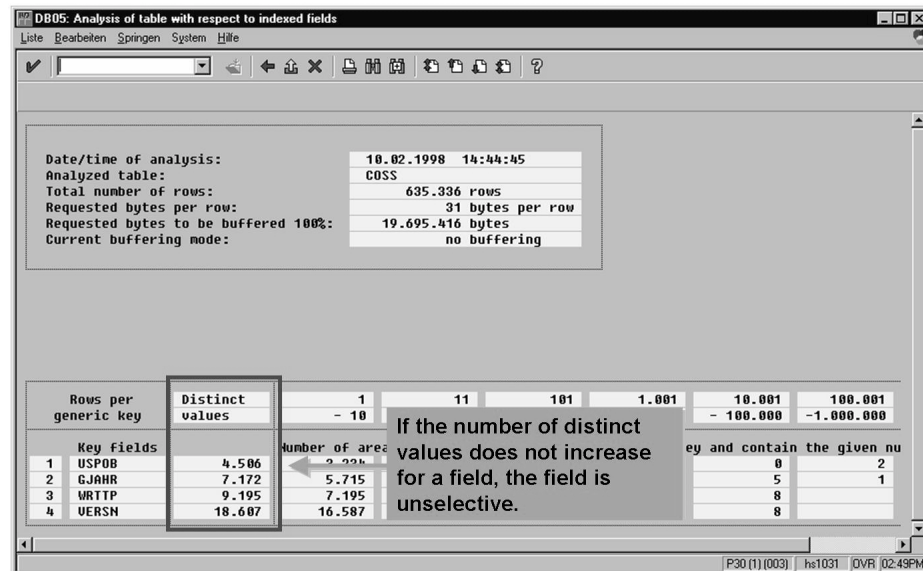


Figure 205: Selectivity Analysis: Number of Distinct Values

The 1 – 10 column displays how many field combinations would return 1 to 10 rows. For a selective index, these numbers are close to the number listed in the *Distinct values* column.

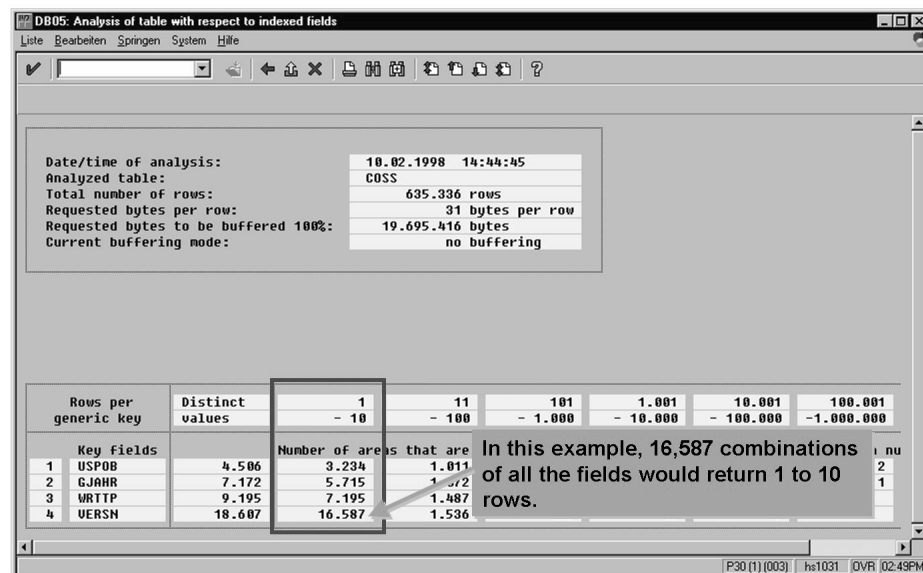


Figure 206: Selectivity Analysis: Number of Rows Returned

In the figure below, eight combinations of all the index fields have a hit set of between 10,001 and 100,000 records. If one of these combinations is specified, an expensive unselective range scan will occur.

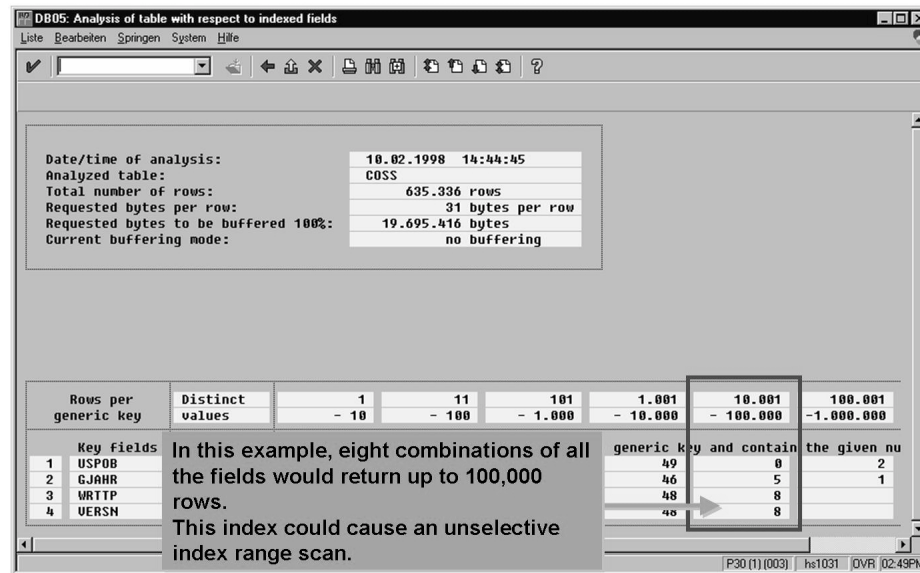


Figure 207: Selectivity Analysis: Inefficient Combinations

In general, an index is not effective if it returns many rows. To determine if you should create an index, even if unselective field combinations exist, you must know the table logic.

Creating and Maintaining Indexes

Indexes are created and maintained in ABAP Dictionary maintenance, which can be accessed using transaction SE11. To display the table fields, enter a table name and select *Display*. The following screen, *Dictionary: Display Table*, shows the fields of the selected table. The primary index fields of the table are marked in the *Key* column. Selecting *Indexes* shows the associated secondary indexes.

Creating a new secondary index:



1. Enter the name of a table in transaction SE11 and choose *Display*.
2. Choose *Indexes* and then select *Create Index*.
3. In the resulting screen, enter the name of the index, a short description, and the name of index fields. Choose *Save*. The index now exist in the ABAP Dictionary, but has not yet been created in the database.
4. To activate the index, choose *Index* → *Activate*.



Caution: The process of activating an index for a large table in the database is especially time-consuming. During this process, INSERT, UPDATE, and DELETE operations affecting the corresponding table are blocked. Therefore, avoid creating new indexes for a large table during business hours. To activate indexes by an appropriate background job, use “ABAP Dictionary: Database Utility” (transaction SE14).



Hint: After activating a new index, you may need to generate new table access statistics so the optimizer can consider the new index when calculating the execution plan.

Before creating an index:



- Check for missing indexes.
- Check if the table statistics are current.
- Check the current table size and how large the table was at the time of the last update statistics.

Missing Indexes

Although an index may be defined as a database index in the SAP system, it may be (or become) missing in the database – for example, because it was not activated or because it was deleted and not recreated during database reorganization. This type of index is called a **missing index**. To determine whether a database index is missing, use the DBA Cockpit and choose *Diagnostics* → *Missing Tables and Indexes*.



Note: In SAP systems where the DBA Cockpit is not available (prior to SAP NetWeaver 7.0) you can use the *Database Performance: Tables and indexes* monitor (transaction DB02) and choose “Missing indexes”.

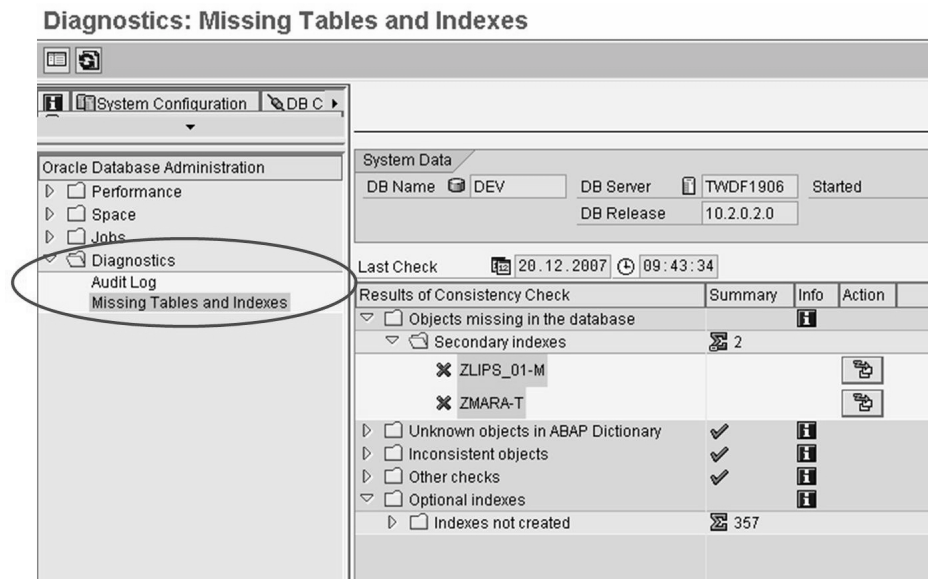


Figure 208: Missing Indexes

In the resulting display, missing primary and secondary indexes are listed separately.

Missing primary indexes require the urgent attention of the database administrator. If a primary index is missing, duplicate keys may be written, and therefore the consistency of the data is no longer guaranteed. Additionally, the lack of a primary index leads to inefficient database accesses, causing performance problems for large tables.

Missing secondary indexes can cause performance problems if the index belongs to a table that is larger than 1 MB. To create the index that was missing:



1. Use transaction DB02 and choose *Diagnostics* → *Missing Tables and Indexes*.
2. In the *Results of Consistency check* menu, choose *Objects missing in the database* → *Secondary indexes*.
3. Select the missing secondary index.
4. Choose *Create on DB*.

To solve the problem of a missing primary index that is known to the ABAP Dictionary but does not yet exist (or no longer exists) on the database, recreate that index in ABAP Dictionary maintenance.

1. Enter the name of a table in transaction SE11 and choose *Display*.
2. Choose *Utilities* → *Database Utility*.
3. In the following screen, select the primary index, then select *Choose*.
4. Chose *Create Database Index*.

If no errors occur, the index has been created on the database. If errors occur – for example, due to duplicate keys – contact SAP.

Exercise 20: Creating an Index

Exercise Objectives

After completing this exercise, you will be able to:

- Decide which index should be created
- Create an index in the SAP system
- Check for missing indexes

Business Example

You have detected performance problems in your system that can be solved by creating an index. You need to know how to check which index should be created and create it.

Task 1:

Execute report **ZZSELECT_##** and analyze the SQL trace.

1. Execute report **ZZSELECT_##** one more time, before starting the trace function of ST05 in another screen.

Result

You will find that table **ZLIPS_##** is accessed using the primary index. This index access is not the most efficient way of accessing this table for this statement.

Task 2:

Analyze the conditions in the WHERE part of the SELECT statement in report **ZZSELECT_##** using transaction ST05.

1. Switch to the ST05 screen (in another window) and analyze the trace again.

Task 3:

The execution of report **ZZSELECT_##** could be accelerated using an appropriate index. Determine possible selective fields to create a new index on table **ZLIPS_##**.

1. Determine the possible field values and the respective value combinations using transaction DB05.

Continued on next page

Result

Analyzing the selectivity for the fields in the WHERE part show that the *MATNR* field is very selective and the other fields do not increase the selectivity.

Task 4:

Create a new selective index for table **ZLIPS_##** and check if this accelerates the execution of report **ZZSELECT_##**.

1. Use transaction SE11 to create a new index, **ZLIPS_##~M**, for table **ZLIPS_##**.
2. Execute your report, **ZZSELECT_##**, one more time before starting the trace function of ST05 in another screen.

Result

You will find that table **ZLIPS_##** is accessed by the new index. This index access is more effective than the access using the primary index.

Task 5:

Delete the new index, **ZLIPS_##~M**, in the database only and select for missing indexes.

1. Use transaction SE14 to delete index **ZLIPS_##~M** in the database.
2. Check for missing indexes.

Task 6:

As a prerequisite for the following lesson, delete the new index, **ZLIPS_##~M**, in the database and in the SAP system.

1. Use transaction SE11 to delete index **ZLIPS_##~M** for table **ZLIPS_##**.

Solution 20: Creating an Index

Task 1:

Execute report **ZZSELECT_##** and analyze the SQL trace.

1. Execute report **ZZSELECT_##** one more time, before starting the trace function of ST05 in another screen.
 - a) Switch to the ST05 screen (in another window) and start the trace.
 - b) Execute report **ZZSELECT_##** again.
 - c) Stop the trace recording. Analyze the trace recording. Determine the duration for the execution of the report and the type of database access used by **ZZSELECT_##**.

Result

You will find that table **ZLIPS_##** is accessed using the primary index. This index access is not the most efficient way of accessing this table for this statement.

Task 2:

Analyze the conditions in the WHERE part of the SELECT statement in report **ZZSELECT_##** using transaction ST05.

1. Switch to the ST05 screen (in another window) and analyze the trace again.
 - a) The conditions in the WHERE part are listed in the trace list. Use the trace function, *Display Trace*, in ST05

The fields in the WHERE part are: MANDT, MATNR, WERKS, and LGORT.

Continued on next page

Task 3:

The execution of report **ZZSELECT_##** could be accelerated using an appropriate index. Determine possible selective fields to create a new index on table **ZLIPS_##**.

1. Determine the possible field values and the respective value combinations using transaction DB05.
 - a) Switch to transaction DB05.
 - b) Enter the name of the table (**ZLIPS_##**), select *Analysis for specified fields*, and remove the *Submit analysis in background* flag.
 - c) Start the analysis for different combinations of possible fields.

Result

Analyzing the selectivity for the fields in the WHERE part show that the **MATNR** field is very selective and the other fields do not increase the selectivity.

Task 4:

Create a new selective index for table **ZLIPS_##** and check if this accelerates the execution of report **ZZSELECT_##**.

1. Use transaction SE11 to create a new index, **ZLIPS_##~M**, for table **ZLIPS_##**.
 - a) Switch to transaction SE11. Enter the name of the table (**ZLIPS_##**) and choose *Change*.
 - b) Select *Indexes* and then choose *Create Index* in the following screen.
 - c) In the next screen, enter the name of the index (**M**), a short description, name of index fields (**MANDT** and **MATNR**), and choose *Save*. The index now exist in the ABAP Dictionary, but has not yet been created in the database.
 - d) To activate the index in the database, choose *Index → Activate*.

Continued on next page

2. Execute your report, **ZZSELECT_##**, one more time before starting the trace function of ST05 in another screen.
 - a) Switch to the ST05 screen (in another window) and start the trace.
 - b) Execute your report, **ZZSELECT_##**, once more. Stop the trace recording.
 - c) Analyze the trace. Determine the duration for the execution of the report and the type of database access used by **ZZSELECT_##**

Result

You will find that table **ZLIPS_##** is accessed by the new index. This index access is more effective than the access using the primary index.

Task 5:

Delete the new index, **ZLIPS_##~M**, in the database only and select for missing indexes.

1. Use transaction SE14 to delete index **ZLIPS_##~M** in the database.
 - a) Switch to transaction SE14.
.
 - b) Enter the name of the table, **ZLIPS_##**, in the *Obj. name* field.
 - c) Choose *Indexes...* and choose the index **ZLIPS_##~M**.
 - d) Choose *Delete database index*.
2. Check for missing indexes.
 - a) Switch to transaction DBACOCKPIT and choose *Diagnostics → Missing Tables and Indexes*. Then choose *Refresh*.
 - b) After this check finished, the number of missing indexes is displayed in the *Objects missing on database* field.
 - c) To recreate your index, select index **ZLIPS_##~M** and choose *Create in database*. In the next screen, select *Direct Processing* and choose *Create*.

Continued on next page

Task 6:

As a prerequisite for the following lesson, delete the new index, **ZLIPS_##~M**, in the database and in the SAP system.

1. Use transaction SE11 to delete index **ZLIPS_##~M** for table **ZLIPS_##**.
 - a) Switch to transaction SE11.
 - b) Enter the name of the table (**ZLIPS_##**) and choose *Change*.
 - c) Choose *Indexes*. Select index **ZLIPS_##~M** in the following screen and choose *Delete*.



Lesson Summary

You should now be able to:

- Create an index
- Figure out what kind index should be created
- Check for missing indexes



Unit Summary

You should now be able to:

- Describe what an index is
- Explain how an index is structured
- Discuss how data is accessed using an index
- Create an index
- Figure out what kind index should be created
- Check for missing indexes



Test Your Knowledge

1. The rows in a table are sorted by the primary key.
Determine whether this statement is true or false.
 - ☐ True
 - ☐ False

2. Which of the following are basic rules of secondary index design?
Choose the correct answer(s).
 - ☐ A Include as many fields as possible.
 - ☐ B Include only selective fields.
 - ☐ C Create as few indexes per table as possible.
 - ☐ D Create as many indexes per table as possible.



Answers

1. The rows in a table are sorted by the primary key.

Answer: False

The data inserted in a table is generally not sorted.

2. Which of the following are basic rules of secondary index design?

Answer: B, C

Unit 10

Cost Based Optimizer

Unit Overview

This unit provides an overview over the concept of database statistics and how the cost based optimizer works. Comprehension of these topics facilitates the evaluation of table access pathes chosen by the cost based optimizer.



Unit Objectives

After completing this unit, you will be able to:

- Name the different types of database statistics
- Describe the update statistics strategy recommended by SAP
- Detect problems with optimizer statistics
- Describe the different factors that influence estimated cost

Unit Contents

Lesson: Update Statistics.....	576
Exercise 21: Detecting Problems with Database Statistics	591
Lesson: Appendix: Cost Evaluation	596
Exercise 22: The Cost-Based Optimizer in Action	605

Lesson: Update Statistics

Lesson Overview

This lesson introduces database statistics. You will learn about the types of statistics and the available methods for creation of database statistics. In addition, we will discuss the recommended strategy for creating database statistics.



Lesson Objectives

After completing this lesson, you will be able to:

- Name the different types of database statistics
- Describe the update statistics strategy recommended by SAP
- Detect problems with optimizer statistics

Business Example

You have detected performance problems in your system, which are caused by optimizer statistics that are too old or not accurate enough. You want to know how to create optimizer statistics. You also need to gather more information about the strategy recommended by SAP to collect statistics.

Fundamentals of Database Statistics

The cost-based optimizer requires database statistics in order to access the required data using the most suitable access path for SELECT, UPDATE, and DELETE statements. Possible access paths are index access or a full table scan, to name a few.

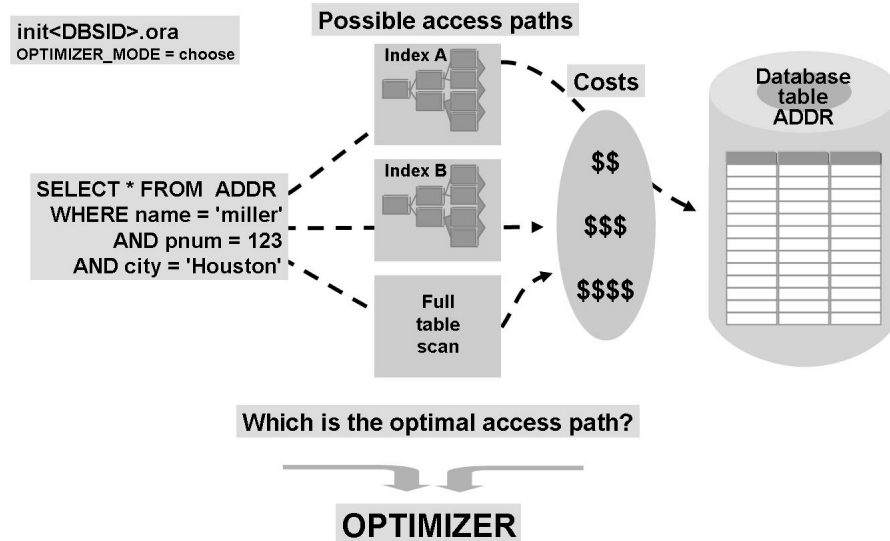


Figure 209: Oracle Cost-Based Optimizer

The cost based optimizer chooses the access path with the fewest expected I/O operations. The optimizer's decision is based on estimates. Whether the estimates are correct depends on the quality of the database statistics. Therefore, the cost-based optimizer requires database statistics.

Database statistics are not required in the following cases if you are using Oracle databases <= 9i:

- When using the **rule-based optimizer** or if an explicit **RULE hint** is specified
- For **SAP table pools and table clusters**
- For **exceptions** according to **DBSTATC**
- For **Oracle DDIC objects**
- For **INSERT statements**, since the inserts carried out are based on free-list or bitmap entries



Caution: Database statistics are required, in general, as of Oracle 10g .

There are different types of statistics available: table, index, and column statistics.

Table statistics

Table statistics contain information such as the number of rows (NUM_ROWS), number of blocks used (NUM_BLOCKS), accuracy (SAMPLE_SIZE), or date of last statistics creation (LAST_ANALYZED). The information is retrieved from the DBA_TABLES view.

Index statistics


Index statistics contain information such as the size of the index tree (BLEVEL), number of leaf blocks (LEAF_BLOCKS), number of distinct keys (DISTINCT_KEYS), key figure for assigning index vs. table (CLUSTERING_FACTOR), accuracy (SAMPLE_SIZE), or date of last statistics creation (LAST_ANALYZED). The information is stored in the DBA_INDEXES view.

Column Statistics

Column statistics contain information such as the number of different values (NUM_DISTINCT), lowest value (LOW_VALUE), highest value (HIGH_VALUE), accuracy (SAMPLE_SIZE), or date of the last statistics creation (LAST_ANALYZED). The information is stored in the corresponding Oracle view DBA_TAB_COLUMNS.

All three statistics types are included in the general term **database statistics** and all types should be generated during statistics creation. For instance, there is no sense in having table statistics without index statistics.

Histogram information, another type of database statistics, is stored in the Oracle view DBA_TAB_HISTOGRAMS. Histograms simply describe the distribution of column values between the lowest existing value (LOW_VALUE) and the highest existing value (HIGH_VALUE). Histograms are optional and represent more detailed column statistics. With no histograms available, the Oracle database assumes an equal distribution of the values inside a column. However, Oracle database histograms are not generally recommended in SAP environments.

 **Note:** For more information, see SAP Note 797629.

Creating Database Statistics

Database statistics need to be created because they are required by the cost-based optimizer. Furthermore, the statistics should be updated periodically. Old statistics might seduce the optimizer to take an inappropriate access path. “Old” in this context is not just a matter of an outdated time stamp, but rather a matter of inaccuracy. Even a five-year-old statistics might be accurate for a given table as long as the table has not been changed to large extent. In contrast, a one-day-old statistics might be outdated because a client was deleted.

The database administrator has several tools at his or her disposal for creating database statistics, including both Oracle tools and SAP tools.

Oracle tools and commands used for creating database statistics:



SQL ANALYZE TABLE

The command ANALYZE TABLE, with appropriate options given, creates a new database statistics. An example would be:

```
ANALYZE TABLE ESTIMATE STATISTICS
SAMPLE 10 PERCENT
FOR TABLE FOR ALL INDEXES FOR ALL INDEXED COLUMNS
SIZE 1;
```

The new statistics is based on an estimate, not an exact computation, covering 10% of the table content. The FOR conditions specify table, index, and column statistics to be created. SIZE 1 means that no detailed histograms will be generated.

Oracle DBMS_STATS package

The DBMS_STATS package provides functionality for displaying, adapting, transferring, and gathering table and index statistics. DBMS_STATS is a more recent method for creating statistics that ANALYZE is expected to replace in the long term.

The following table lists the pros and cons of DBMS_STATS compared with the ANALYZE command.

Pros	Cons
Usually a shorter runtime	Only cost-based-optimizer-relevant statistical data is determined; for monitoring purposes, you still need the ANALYZE-based statistics.
Table-internal parallel processing is possible (see SAP Note 408532)	No statistics for Oracle cluster tables
Correct statistics even for columns with identical character strings in the first 32 characters (see SAP Note 365480)	Oracle 8i: Histograms cannot be created when you use parallel processing.
Statistics can be transported to other systems, for example, from the productive to the test system	
Backup and reactivation of statistics is possible	

Unfortunately, the statistics generated by the DBMS_STATS package and the ANALYZE command are not fully compatible, which causes problems when you create DBMS_STATS statistics for objects with ANALYZE statistics, and vice versa.

When changing the method, you should therefore first delete statistics using the function of the method used to date (ANALYZE TABLE ... DELETE STATISTICS or DBMS_STATS.DELETE_..._STATS) before new statistics are created. Deletion of these old statistics has already been implemented in current versions of BRCONNECT.



BR*TOOLS or BRCONNECT

BR*TOOLS is the recommended method for creating new database statistics. BR*TOOLS calls BRCONNECT with the right settings. Alternately, you could also directly call BRCONNECT, but then you must specify the command options manually. Updates of database statistics using BRCONNECT commands should be scheduled regularly when using Oracle 10g, for example, on a daily basis. The database statistics updates can be scheduled using the DBA Cockpit (transaction DB13 or DBACOCKPIT).

Transaction DB20

This transaction allows you to create new database statistics for selected tables. The new statistics automatically comprises all components, that is, table, index, and column statistics.

Report RSANAORA

Using this report, you can also generate complete new statistics for a given table. Alternately, you can also just create a new index statistics of a selected index.



Note: As of Oracle 10g, by default, BRCONNECT employs the DBMS_STATS package to create statistics, but you can switch to ANALYZE TABLE if you want to. Please consider that the recommended strategy for creating statistics is to use the DBMS_STATS package. To change the statistics method used by BRCONNECT, you must edit the initialization profile init<DBSID>.sap.



Caution: When using Oracle 9i or lower, BRCONNECT uses the ANALYZE TABLE command by default. Database statistics updates using BRCONNECT should be scheduled on a weekly basis.

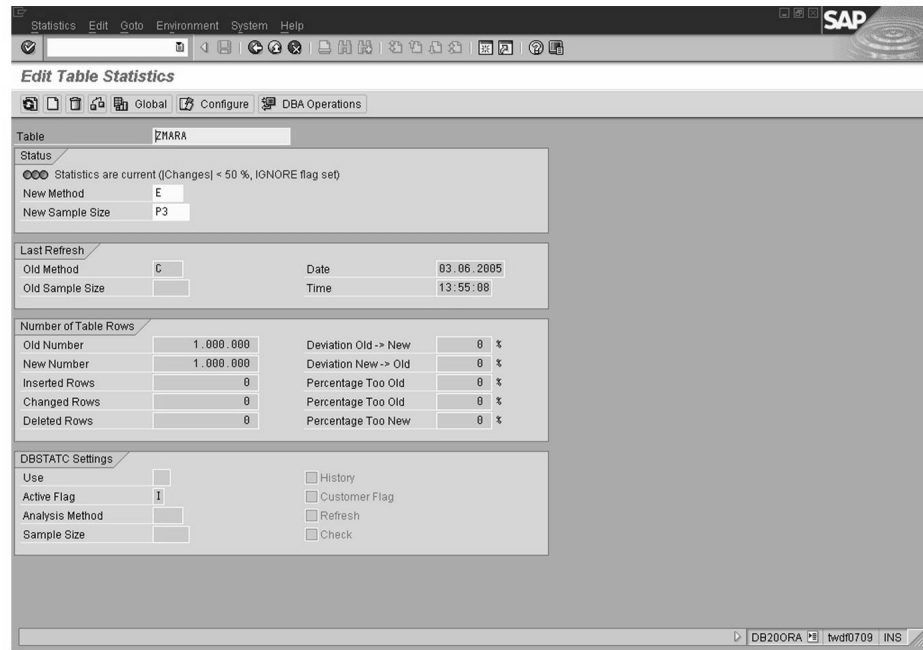


Figure 210: DB20: Edit Table Statistics

Recommended Strategy for Creating Database Statistics

SAP recommends a regular creation of database statistics. Keep in mind that inaccurate database statistics, commonly known as old statistics, might lead the cost-based optimizer to an inappropriate access path. As a consequence, you might observe expensive SQL statements and long dialog response times.

In production operation, SAP strongly recommends that you update statistics daily using the DBA Planning Calendar (transaction DB13). If you use a tool such as cron (UNIX) or at (Windows), we recommend the following standard command:

```
brconnect -u / -c -f stats -t all.
```

SAP recommends that you use the above method to update statistics, and also recommends you to use this call immediately after an SAP system upgrade or after a large amount of data has been changed in the database. In the above call, BRCONNECT checks statistics for all SAP tables and indexes using the standard method. If the change threshold for a particular table is exceeded, statistics are updated for that table.

BRCONNECT performs update statistics in one phase using a two-step approach. BRCONNECT checks each table first to see if the statistics are out of date and then, if required, updates the statistics on the table immediately after the check.

Creating statistics does not block any database objects. In theory, therefore, you can create statistics while the system is running. However, since the creation of statistics involves an additional CPU and I/O load, it should be carried out during a period of minimal workload if at all possible. The BRCONNECT two-step strategy to collect statistics is necessary due to performance reasons. Just creating new statistics generally for all tables is not feasible. The creation of new statistics information must be restricted to the necessary minimum set of tables with old statistics.

The creation of a new statistics is based on a 50% criterion: New table statistics will be created if the table content, that is, the number of rows, has changed at least by 50%. Both an increase or a decrease of table content is taken into account.

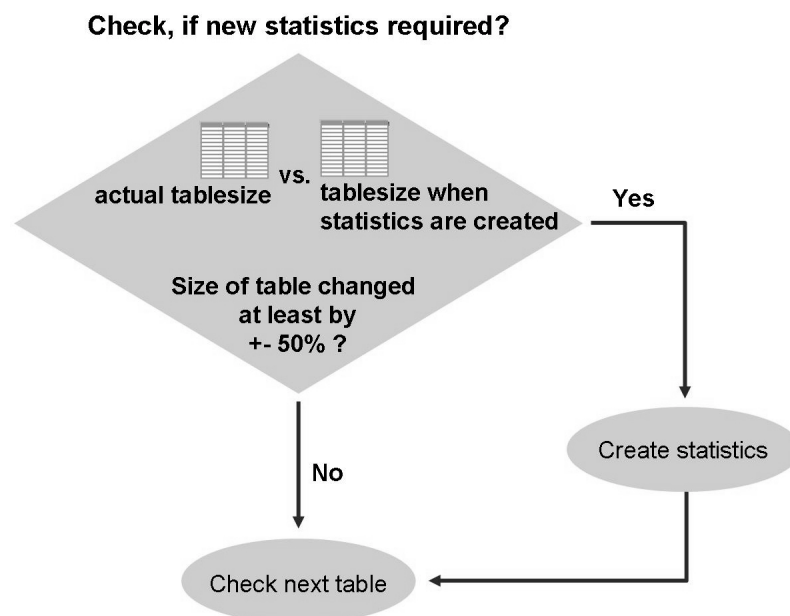


Figure 211: Deciding Whether a New Table Statistics is Needed



Caution: If new columns or indexes are created (even in transports), the table statistics must be updated (for instance, using transaction DB20). Other functions, such as DBA Cockpit, BRCONNECT without special force options, or the sole analysis of an index (whether newly created or reorganized), can cause incomplete statistics and then cause considerable performance problems. As of Oracle 10g, Oracle automatically creates index statistics when an index is created. This means that you do not need to also create the statistics manually.

Refer to SAP Note 657536 for more information.

Activation of Table Monitoring

Starting with BRCONNECT 6.10, you can activate table monitoring, which is recommended by SAP. This monitoring supports the statistics check by using information about table changes. Activation of table monitoring speeds up the check phase of the update statistics significantly. In addition, tables that were created dynamically by the SAP system, which is often the case in BW systems, are handled automatically by BRCONNECT during the next run.



Note: For more details, see SAP Notes 408527 and 628590.

New Statistics with Oracle 10g

As of Oracle database 10g, the rule-based optimizer is no longer supported. In Oracle 9i or lower, due to performance reasons, no statistics were calculated for the Oracle Dictionary. The execution plan to access dictionary tables was determined by the rule-based optimizer.

Because the rule-based optimizer is no longer supported in Oracle 10g, you are required to create statistics for the Oracle Dictionary. In addition, you need to calculate system statistics.

Up to Oracle 9i no statistics were usually created for the schema or users SYS and SYSTEM, and the access optimization to the Oracle Dictionary occurred using the rule-based optimizer. Oracle 10g still only supports the cost-based optimizer; the rule-based optimizer is no longer supported. This means that from now on, accesses to the Oracle Dictionary are only optimized by the cost-based optimizer. Therefore, statistics must also be created for the Oracle Dictionary. This primarily concerns the database schemata SYS and SYSTEM.

Automatic Statistics Creation

Creating and updating statistics (table and index statistics) for the normal user schemata – that is, the SAP schemata SAPR3, SAP<SAPSID> and SAP<SAPSID>DB – occurs in the SAP environment using BR*Tools (to be precise, using BRCONNECT).

As of Oracle 10g, Oracle database statistics can also manage completely automatically. This is implemented in the database as the DBMS_SCHEDULER job GATHER_STATS_JOB. However, this method is not yet supported in the SAP environment, and is therefore deactivated to avoid conflicts with the BRCONNECT method. In new SAP installations as of SAP NetWeaver 7.00, this automatic function is deactivated by SAPINST.

Table Monitoring

To establish which table statistics are no longer exact (stale), you can use the table monitoring feature, which calculates the approximate number of inserts, updates, and deletes for each table (and also takes truncation into account). Table monitoring must be active in the SAP environment. Table monitoring is active in Oracle database 10g as a default if the parameter `STATISTICS_LEVEL` is set to the default value `TYPICAL`. In the `*_TAB_MODIFICATIONS` views, you can see an overview of the number of changes in a table since the statistics were last created.

Oracle Dictionary Statistics

As of BR*Tools 7.00 patch, level 15, the BRCONNECT will support the creation of Oracle Dictionary statistics using the methods described below. We recommend that you replace older versions of BRCONNECT with a newer version.

The relevant call using BRCONNECT is:

```
brconnect -u / -c -f stats -t oradict_stats
```

Using this command ensures that the system creates current, highly accurate statistics for all objects of the Oracle Dictionary.



Note: It is recommended that you create new dictionary statistics when the Oracle Dictionary may have changed considerably, for example:

- After installing a database patch set
- After an SAP upgrade
- If another SAP system has been installed (MCOD)

Otherwise, you should create dictionary statistics once per quarter.

Oracle Fixed Object Statistics

Part of the Oracle Dictionary consists of X\$ tables, which are known as **fixed objects**. Fixed objects are Oracle memory structures that behave similarly to tables but are not stored physically. They are low-level data sets that feed other views with information. These dynamic performance tables contain information about current database activity.



Note: As of Oracle 10g it is strongly recommended that you create fixed object statistics in general because the rule-based optimizer is not supported and the cost-based optimizer needs statistical information to calculate the optimal access path. These statistics should be created at a time of representative database load.

Fixed object statistics can be created using BRCONNECT with the following command:


```
brconnect -u / -c -f stats -t oradict_stats
```

If fixed object statistics are not created properly, you might have long runtimes when accessing Oracle performance data (for example, Oracle session overview in ST04).



Note: It is recommended that you update the fixed object statistics from time to time (for example, every three months) and in case of significant system changes (for example additional application functionality or an Oracle upgrade).

The creation of fixed object statistics can seriously impact the database activities running in parallel. This problem usually shows up in combination with large buffer pools. Therefore it is a good idea to run the fixed object statistics gathering at a time of non-critical system activities.

Oracle System Statistics

System statistics describe the attributes of the hardware of the system, for example, I/O performance or processor rate. These performance figures (in the form of statistics) provide the cost-based optimizer with a more precise calculation of CPU costs and I/O costs and assist in finding a better access plan.

With system statistics, the CPU consumption, single-block reads, and multi-block reads are weighted correctly so that the cost-based optimizer has significantly more data available for the actual resource and time demand, and can thus make more optimal decisions.



Oracle <= 9i

```
SELECT STATEMENT ( Estimated Costs = 4 , Estimated
#Rows = 466 )
```

```
TABLE ACCESS FULL DBSTATC
( Estim. Costs = 4 , Estim. #Rows = 466 )
```

system independent ratio based on
DB_FILE_MULTIBLOCK_READ_COUNT

Oracle 10g

```
SELECT STATEMENT ( Estimated Costs = 6 , Estimated
#Rows = 466 )
```

```
TABLE ACCESS FULL DBSTATC
( Estim. Costs = 6 , Estim. #Rows = 466 )
Estim. CPU-Costs = 295.157 Estim. IO-Costs = 4
```

system dependent ratio based on
System statistics

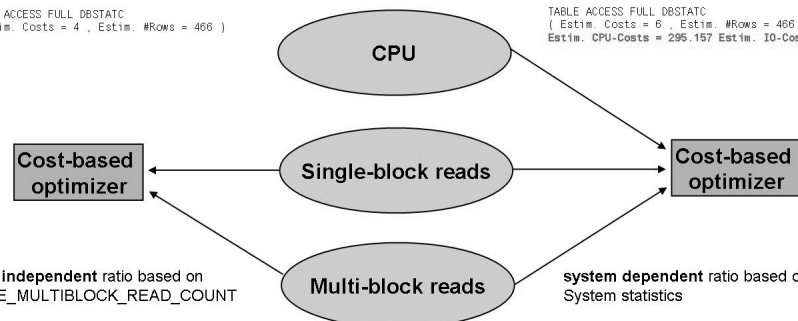


Figure 212: Oracle System Statistics

Oracle distinguishes between WORKLOAD and NOWORKLOAD system statistics:

- NOWORKLOAD statistics contain information that Oracle considers to be workload-independent. Internal default values are used with Oracle 9i; from Oracle 10g the relevant values such as CPU speed when the system is not under load are determined system-dependently.
- WORKLOAD statistics contain additional key figures that may depend on the system load, such as average access time for a single-block read. Since WORKLOAD statistics are load-dependent, it is important to generate these statistics during a characteristic workload phase.



Note: SAP recommends creating NOWORKLOAD system statistics on a regular basis.

The relevant call using BRCONNECT is as follows:

```
brconnect -u / -c -f stats -t system_stats
```

It is recommended to create new system statistics:

- Initially after creating the database
- After creating new tablespaces
- After a database upgrade to Oracle database 10g
- When you change the hardware of the database server (other/more CPUs, more hard disks, or faster network)
- Once a quarter

Detecting Problems with Database Statistics

Problems with database statistics essentially belong to three groups: missing statistics, outdated statistics, and inaccurate statistics.

In the first case, **missing statistics**, two subtypes can be distinguished: no statistics at all or missing index or column statistics. You can use transaction DB20 to check the existence of a database statistics for a specific table. If necessary, create statistics for the given table using transaction DB20. Otherwise, schedule the database statistics run in transaction DB13.

Detecting missing parts of a statistics, like **missing index or column statistics**, is more difficult. To find tables with missing index or column statistics, special SQL scripts must be executed. For details, see SAP Note 588668.

Problems with **outdated statistics** for a single table can also be identified and fixed with transaction DB20. Otherwise, schedule the regular database statistics update run in DB13.

Even more subtle are problems due to an **inaccurate sampling** size or method used for statistics generation. The accuracy of new statistics depends on the size of the table. Based on the following thresholds, BRCONNECT computes or estimates the new database statistics:

- Fewer than 10,000 rows: Exact calculation
- Fewer than 100,000 rows: ESTIMATE with 30% of the table
- Fewer than 1,000,000 rows: ESTIMATE with 10% of the table
- Fewer than 10,000,000 rows: ESTIMATE with 3% of the table
- More than 10,000,000 rows: ESTIMATE with 1% of the table

The database administrator can always specify tighter values for the generation of database statistics than those used by BRCONNECT. However, keep in mind that more accurate statistics require more time. During that elongated time period, the table might not be used productively.

Using the following SQL script, the database administrator can check whether table statistics exist whose thresholds might be too loose.

```
SELECT * from DBA_TABLES
WHERE NUM_ROWS > 500000 AND
SAMPLE_SIZE < 0.5 * 0.1 * NUM_ROWS;
```

The script focuses on tables with more than 500,000 table entries. For this category, the new table statistics should be based on a 10% estimate. The script then looks for tables where the sampling size is lower than 50% of the recommended threshold. In principle, you could choose other search options.

Handling Unfavorable Decisions by the Cost-Based Optimizer

In certain situations, the cost-based optimizer may make unfavorable decisions, although there are no cost-based optimizer errors, for example:

- Tables with heavily fluctuating volumes of data
- Selective conditions on columns that do not have many attributes

Creating new statistics always involves the risk that individual accesses may be compromised. In this case, it is possible to retrieve the previous statistics. This can be done as of Oracle 10g by using the DBMS_STATS package.

In Oracle 9i or lower, performance problems were solved by activating the rule-based optimizer by using the RULE hint, or by not creating cost-based optimizer statistics. However, this is no longer possible, since the rule-based optimizer is no longer

supported in Oracle 10g. In addition, with Oracle 9i, there are already situations in which the rule-based optimizer cannot be used (for example, in the case of FIRST_ROWS hint).

In Oracle 10g SAP delivers dummy statistics to solve these problems. These dummy statistics are delivered for a few tables where it is known that the cost-based optimizer makes incorrect decisions. Therefore you have to implement a script for the automatic postprocessing of critical statistic values, which is attached to SAP Note 1020260. This script adjusts individual statistic values, excludes the relevant tables from the statistics creation using the BRCONNECT tool (ACTIV=I in DBSTATC), and locks the statistics at Oracle level (DBMS_STATS.LOCK_TABLE_STATS).



Hint: As of Oracle 10g, Oracle keeps old statistics for 31 days by default to maintain the option of being able to activate such statistics using DBMS_STATS.RESTORE_TABLE_STATS. In particular cases, the history statistics can lead to a massive memory requirement in the SYSAUX tablespace.

Manual Fine Tuning of Computed Database Statistics

The manual adjustment of database statistics should be regarded as a last resort. First, try other methods, like creation of new or more accurate database statistics or adapting your ABAP statement.

Only adjust the statistics after you have performed a thorough check. In a non-production environment, check whether the changed statistics have unwanted side effects before you transfer the changes to the production system. Follow a least-change strategy; abrupt changes might have severe impacts. To tune a statistical value, you should perform a large number of tests with different values.

To keep the changed statistics, you must prevent BRCONNECT from overwriting the statistics during its statistical runs. For this purpose, create an entry with ACTIV=I in the DBSTATC table for the relevant table.

Manual adjustment must be repeated after new regular statistics have been created.

Optimizer Decision

The following picture shows the roadmap of selecting the optimizer and estimating the preferred access path.



Decision of the Optimizer

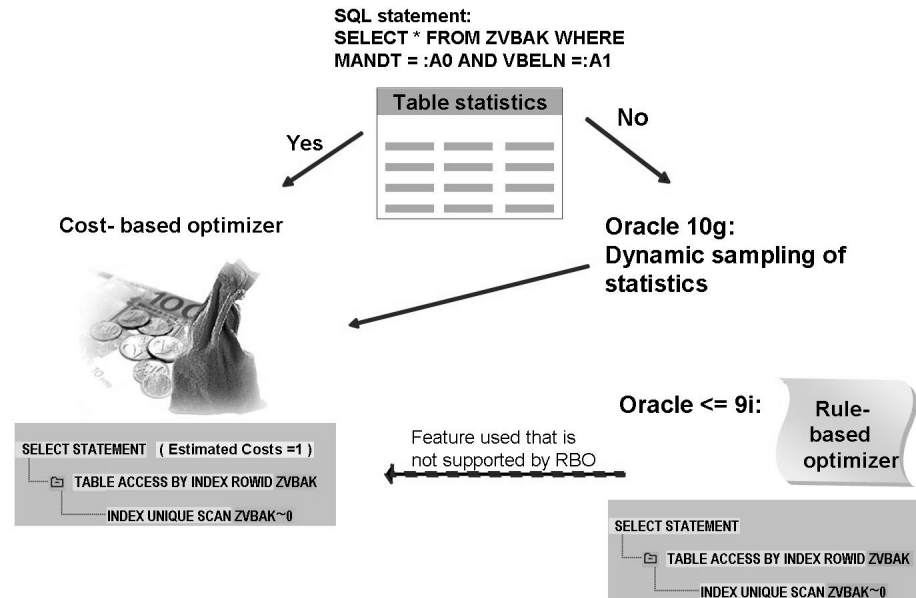


Figure 213: Roadmap: Selection of Database Optimizer

As of Oracle 10g, dynamic sampling is used as standard if there are no statistics for a table. Dynamic sampling generates statistics “on the fly” on the basis of a small sample, and then the cost-based optimizer is used. For this reason, the cost-based optimizer is used in Oracle 10g even though there are no statistics.

The result of the sampling may lead to various execution plans. Ensure that ALL tables have statistics with Oracle 10g to avoid dynamic sampling.

Exercise 21: Detecting Problems with Database Statistics

Exercise Objectives

After completing this exercise, you will be able to:

- Monitor and create new database statistics

Business Example

You identified expensive SQL statements in your SAP system. Now you want to check whether the problems are due to inaccurate database statistics.

Task 1:

Use the DBA Cockpit and look for tables with inaccurate database statistics.

1. Call transaction DBACOCKPIT.
2. Open the SQL Command Editor sub-monitor.
3. Using SQL, search for tables with (a) less than 750000 entries, and (b) a statistics sample size 50% lower than the recommended default value.

Result

You get a list of tables displayed; among them is table ZLIPS_##.

Task 2:

Display the database statistics for table ZLIPS_##.

1. Call transaction DB20.
2. Select table ZLIPS_##.

Result

Transaction DB20 shows that statistics are missing for table ZLIPS_##.

Task 3:

Create new database statistics for table ZLIPS_## using the recommended settings.

1. You are still in transaction DB20 and can see the detailed information on table ZLIPS_##. Specify the method **Compute statistics**.

Continued on next page

2. Create new statistics.

Result

After the update finishes, you have successfully created an accurate database statistics for table ZLIPS_##. According to the table size of 100,000 entries, a sample size of 10% is recommended now.

Solution 21: Detecting Problems with Database Statistics

Task 1:

Use the DBA Cockpit and look for tables with inaccurate database statistics.

1. Call transaction DBACOCKPIT.
 - a) See task description.
2. Open the SQL Command Editor sub-monitor.
 - a) Choose *Performance* → *Additional functions* → *SQL Command Editor*.
3. Using SQL, search for tables with (a) less than 750000 entries, and (b) a statistics sample size 50% lower than the recommended default value.
 - a) Execute the following SQL statement:

```
Select * from DBA_TABLES
Where NUM_ROWS < 750000 AND
SAMPLE_SIZE < 0.5 * 0.1 * NUM_ROWS;
```

Result

You get a list of tables displayed; among them is table ZLIPS_##.

Task 2:

Display the database statistics for table ZLIPS_##.

1. Call transaction DB20.
 - a) See task description.
2. Select table ZLIPS_##.
 - a) Enter ZLIPS_## in the *Table* input field and refresh the display.

Result

Transaction DB20 shows that statistics are missing for table ZLIPS_##.

Continued on next page

Task 3:

Create new database statistics for table ZLIPS_## using the recommended settings.

1. You are still in transaction DB20 and can see the detailed information on table ZLIPS_##. Specify the method **Compute statistics**.
 - a) Enter **C** in the *New Method* field.
2. Create new statistics.
 - a) Choose *Create statistics*.

Result

After the update finishes, you have successfully created an accurate database statistics for table ZLIPS_##. According to the table size of 100,000 entries, a sample size of 10% is recommended now.



Lesson Summary

You should now be able to:

- Name the different types of database statistics
- Describe the update statistics strategy recommended by SAP
- Detect problems with optimizer statistics

Lesson: Appendix: Cost Evaluation

Lesson Overview

This lesson discusses cost evaluation in an Oracle database.



Lesson Objectives

After completing this lesson, you will be able to:

- Describe the different factors that influence estimated cost

Business Example

Due to performance problems in your system, which are caused by optimizer statistics, you have implemented the strategy recommended by SAP to collect statistics. To increase your knowledge about the Oracle cost-based optimizer, you need more information about the main factors that are considered by the optimizer to calculate the estimated costs for an access path.

Introduction

The Oracle cost-based optimizer uses an internal algorithm to calculate the costs for the possible access paths (for instance, full table scan or index range scan), to determine the optimal access. The blocks to be read for the access (that is, the I/O costs) can be used to measure the calculated costs. As of Oracle 10g, SAP recommends that you create NOWORKLOAD system statistics, whereby the calculated CPU costs are included in the cost calculation. The calculation of CPU costs is based on internal assumptions about the cost of certain execution steps and is therefore very hard to follow. For this reason, this lesson concentrates on I/O-related cost calculation. Furthermore, it only analyzes costs as regards conditions with bind variables, since this is the standard case in SAP environments.

Optimizer-Related Oracle Parameters

The following Oracle system parameters influence the cost evaluation in an SAP environment. Some parameters directly go into the cost evaluation as multiplication factors. Other parameters, mostly memory-related ones, affect the overall performance of the Oracle database.

Parameter OPTIMIZER_MODE

As of Oracle 10g, do not set this parameter. The default parameter value will be used.



Hint: For Oracle 9i and lower, please consider the following. As of SAP R/3 4.0, this parameter must be set to **CHOOSE**, which activates the cost-based optimizer.

Parameter OPTIMIZER_INDEX_COST_ADJ

The value of this parameter is a multiplication factor that reduces the cost of index accesses down to the specified value in percent. For instance, a value of 10 reduces the cost of the index access by 90%: $\text{cost_new} = \text{cost_old} * 0.1$. The parameter simply acts as a weighting factor for index accesses.

Parameter OPTIMIZER_INDEX_CACHING

This parameter specifies how many percent of the index blocks should be cached in memory when executing IN-List iterations or accessing inner tables of nested loop-joins. Larger parameter values might speed up the above-mentioned access paths but might harm other accesses. Values for this parameter vary between 1% and 100%.



Hint: Despite a parameter value of zero, a certain number of index blocks is cached. This number is determined via internal algorithms. If you set the parameter value to zero, the Oracle database might, in some cases, estimate lower costs than for small positive values. In short, it might be worth trying to set the parameter to zero and watching its effect on estimated costs.

Parameter DB_FILE_MULTIBLOCK_READ_COUNT

This parameter affects multi-block read accesses (that is, accesses in which several successive blocks can be imported at once). The costs of scattered reads like full table scans and index fast full scans have a reciprocal relationship to this parameter: the larger the parameter, the smaller the costs.

The parameter DB_FILE_MULTIBLOCK_READ_COUNT affects the calculation of costs under the following circumstances:

- Oracle 10g or higher without WORKLOAD system statistics and in which the parameter is set explicitly
- Oracle 9i or earlier without WORKLOAD system statistics



Note: In Oracle 9i and earlier, this parameter should be fixed to 8 for transactional SAP systems. If the calculation process uses this parameter, its value must be replaced by 6.59. The parameter DB_FILE_MULTIBLOCK_READ_COUNT is only approximately included in the calculation. For the cost calculation, this parameter is divided by a correction factor.

Parameter _DB_FILE_OPTIMIZER_READ_COUNT

In Oracle 10g and higher, the parameter _DB_FILE_OPTIMIZER_READ_COUNT affects the calculation of costs in cases without WORKLOAD system statistics and where DB_FILE_MULTIBLOCK_READ_COUNT is not set explicitly.

Parallel query

If you use parallel query, actions that can be processed in parallel, such as full table scans, result in a lower calculated costs that corresponds to the parallel processing level.



Note: See SAP Note 651060 for details.

Parameter HASH_AREA_SIZE / PGA_AGGREGATE_TARGET

In the normal SAP R/3 system, hash joins are deactivated by HASH_JOIN_ENABLED=FALSE; in the BW environment, the PGA Memory area available for hash joins (see SAP Note 789011) plays a role in the cost calculation of the cost-based optimizer. The higher the HASH_AREA_SIZE or PGA_AGGREGATE_TARGET is set, the lower the costs for a hash join.

Parameter SORT_AREA_SIZE / PGA_AGGREGATE_TARGET

Using SORT_AREA_SIZE or PGA_AGGREGATE_TARGET determines how much memory can be used for sortings before the temporary tablespace must be accessed (see SAP Note 789011). The higher the SORT_AREA_SIZE or PGA_AGGREGATE_TARGET selected, the lower the costs of sortings as part of operations such as ORDER BY or sort merge.

The above-mentioned Oracle parameters are set either in the init<DBSID>.ora or in the spfile<DBSID>.ora file. They can be displayed in the DBA Cockpit (transaction DBACOCKPIT); choose *Performance* → *Additional Functions* → *Database Parameters*.



Additional Functions - Database Parameters

System Configuration DB

Oracle Database Administration

- Performance
 - Performance Overview
 - Wait Event Analysis
 - SQL Statement Analysis
 - Statistical Information
 - Feature Monitoring
 - Additional Functions
 - SQL Command Editor
 - Display GV\$-Views
 - Display DBA Tables
 - Database Parameters**
 - Alert Log
 - Checkpoints
 - Oracle Net
 - RAC Statistics
 - Space
 - Jobs
 - Diagnostics

Database Parameters

DB Name DEV Started 28.11.2007

DB Server TWDF1906 10:22:15

DB Release 10.2.0.2.0

Active parameters Parameters history SPFILE

Active Parameters

Instance Id	SID	Parameter	Parameter value
1	*	open_links_per_instance	4
1	*	optimizer_dynamic_sampling	2
1	*	optimizer_features_enable	10.2.0.2
1	*	optimizer_index_caching	50
1	*	optimizer_index_cost_adj	20
1	*	optimizer_mode	ALL_ROWS
1	*	optimizer_secure_view_merging	TRUE
1	*	os_authent_prefix	OPS\$
1	*	os_roles	FALSE
1	*	parallel_adaptive_multi_user	TRUE
1	*	parallel_automatic_tuning	FALSE
1	*	parallel_execution_message_size	16384
1	*	parallel_instance_group	

Figure 214: Optimizer-Related Oracle Parameters

Cost Evaluation: An Approximative View

Oracle has not revealed the exact algorithms used for cost evaluation to the public. The following information, although based on thorough investigations, is therefore only approximative.

You must decide whether to use a full table scan or index access. A common misconception is that an index access path is always better than a full table scan. This statement might be true for large application tables, but for small tables, which can be completely buffered, the full table scan should be faster. Nevertheless, many developers and database administrators often try to favor index accesses over full table scans.

The following figure gives a short summary of the approximations of the most important access paths for the cost calculation. The list of formulas in this figure contains several statistics variables.

➔ **Note:** These approximations for the cost calculation are discussed in detail in SAP Note 750631.



Full Table Scan

Oracle 9i:
$$\frac{\text{BLOCKS}(\text{table})}{\text{DB_FILE_MULTIBLOCK_READ_COUNT} * \text{degree of parallelism}}$$

Oracle 10g:
$$\left[\frac{\text{BLOCKS}(\text{table}) * \text{Average duration of a multi block read}}{\text{Average number of blocks that are read with a multi block read}} \right] * \left[\frac{\text{Average duration of a single block read}}{\text{DB_FILE_MULTIBLOCK_READ_COUNT} * \text{degree of parallelism}} \right]$$

Index Unique Scan

+ Table Access by Index ROWID

$$\frac{(\text{BLEVEL}(\text{index}) + 1) * \text{OPTIMIZER_INDEX_COST_ADJ}}{100}$$

$$\frac{(\text{BLEVEL}(\text{index}) + 2) * \text{OPTIMIZER_INDEX_COST_ADJ}}{100}$$

Index Range Scan

+ Table Access by Index ROWID

$$\left(\text{BLEVEL}(\text{index}) + \frac{\text{LEAF_BLOCKS}(\text{index})}{\prod \text{Distinct Values}} \right) * \text{OPTIMIZER_INDEX_COST_ADJ}$$

$$\left(\text{BLEVEL}(\text{index}) + \frac{\text{LEAF_BLOCKS}(\text{index})}{\prod \text{Distinct Values}} + \text{Clustering Factor} * \text{FilterFactors} \right) * \text{OPTIMIZER_INDEX_COST_ADJ}$$

Figure 215: Approximations for the Cost Calculation

The cost-based optimizer statistics are the starting point for calculating costs, in particular, the following variables are taken from cost-based optimizer statistics:

- **BLOCKS(table)** is the number of the table blocks (up to the high watermark)
- **BLEVEL(index)** is the amount of the index (number of levels minus 1)
- **LEAF_BLOCKS(index)** is the number of index leaf blocks
- **CF(index)**: Clustering factor of an index, which measures for the arrangement in the table vs. index. See SAP Note 832343 for details.
- **Distinct values** is the number of different values that appear in a column.
- **Filter factor** are filtering factors. The filtering factors are calculated for the individual components of the WHERE clause. The exact formula of the filtering factor strongly depends on the linkage of the conditions in the clause, that is, whether AND, LIKE, OR, or NOT links are used. For more details, see SAP Note 750631.

In the case of system statistics, the following values are also included in connection with I/O costs:

- Average duration of a single-block read
- Average duration of a multi-block read
- Average number of blocks that are read with a multi-block read

The following figures will give some examples for the cost evaluation.

For a full table scan, the number of blocks allocated is used for the calculation of the estimated costs. Empty data blocks that have never been used do not have to be scanned and hence are not considered.

In the example in the figure below, table ZLIPS_00 contains 100,000 rows in 684 blocks. First, the table has 684 allocated blocks and no empty blocks. The access path with the lowest estimated cost is a full table scan with estimated cost of 117.

Then some table rows were inserted. After a new optimizer statistics update, the table contains 1,000,000 rows. However, now there are 6,858 allocated blocks and no empty blocks. The cost for the full table scan now increases by a factor of 10, which is proportional to the number of allocated blocks.

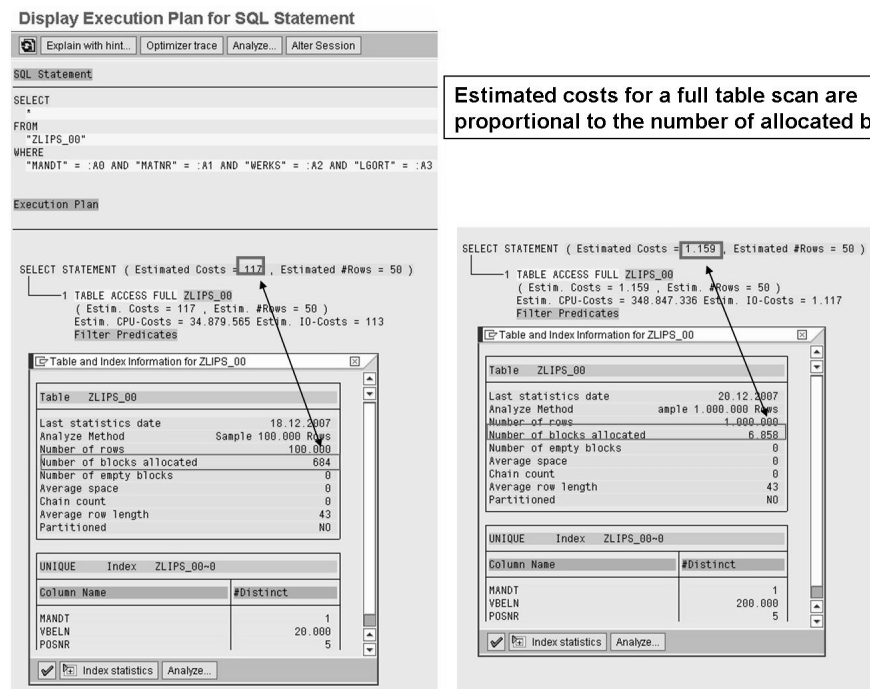


Figure 216: Number of Blocks Read for a Full Table Scan

For an index range scan, the costs calculated depend on several factors:

- The selectivity of the indexed fields
- The efficiency of finding the data in the table (average data blocks per key)
- The number of B*Tree levels
- The number of index range scans that have to be performed
- The operator used in the WHERE condition

The example below shows the costs for an index unique scan.

The costs for one index unique scan are calculated to 1. This number is derived from the number of blocks expected to be read on the index, which is 3, plus the blocks expected to be read on the table, which is 1 in this case. Because parameter OPTIMIZER_INDEX_COST_ADJ is set on the database, the resulting number, which is 4, is then divided by 20, which is 0.2, and then rounded to the next number, which is 1.

The number of blocks that have to be read on the table and index is derived from the selectivity of the indexed fields. For a unique column combination, the number of table columns found for an index combination is always 0 or 1. Therefore, only the index root and branch blocks have to be read, plus one index leaf block and one table block.

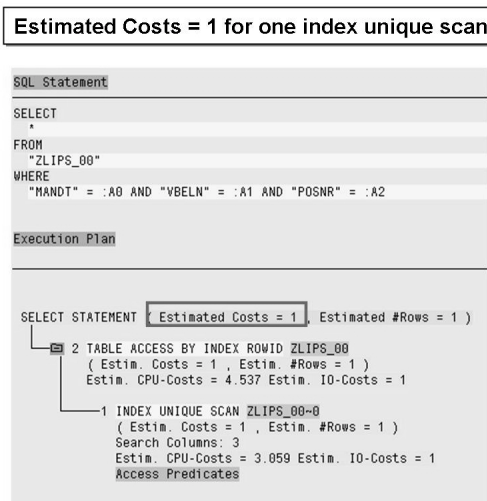


Figure 217: Costs for an Index Unique Scan

Simply looking at the cost formulas does not suffice to deduce which access path will be taken. Instead, the structure of the WHERE clause, which is to be processed, is decisive.

Manual Oracle Hints

Oracle hints are a manual advice to guide the cost-based optimizer in the “right” direction. Starting with SAP R/3 4.5, Oracle hints could even be appended to Open SQL statements. Over time, the importance of hints has largely decreased because the algorithms of the cost-based optimizer have been improved.

There are special hints available. The RULE hint, for instance, redirects to the rule-based optimizer. Access paths determined via the rule-based optimizer have no estimated cost, since the selection of a preferred access path is based on rules, not on database statistics and derived costs.



Note: For more details on hints, see SAP Note 772497.

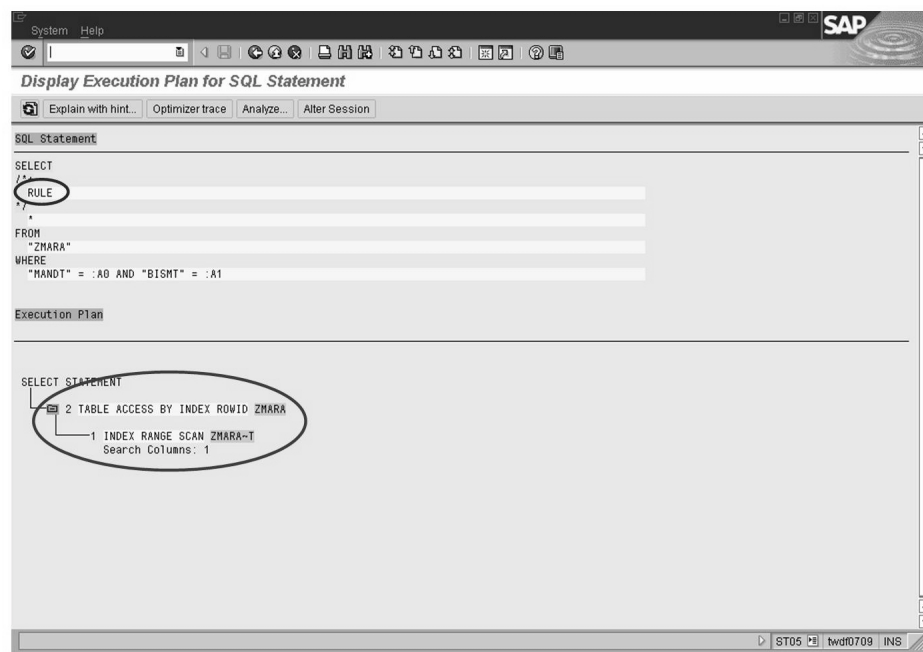


Figure 218: No Estimated Costs When Using the Rule-Based Optimizer

Automatically Generated Oracle Hints

Besides manual hints, there are also generated hints. The use of generated hints can be controlled via **system parameter** `db/ora/use_hints`. Hints for Open SQL statements can be generated and used using the default value of **-1**. Parameter value **0** completely prevents the use of generated hints. For more details, see the parameter documentation by using transaction RZ11.

Exercise 22: The Cost-Based Optimizer in Action

Exercise Objectives

After completing this exercise, you will be able to:

- Interpret the access path chosen by the cost-based optimizer

Business Example

When investigating expensive SQL statements, you would like a closer look at the chosen execution plan. You also want to compare and rationalize the costs of different execution plans.



Hint: This exercise requires two parallel SAP windows open.

Task:

Investigate an expensive SQL statement of your choice. Try to rationalize the access paths chosen by the cost-based optimizer. Are there dependencies of the computed costs on the table statistics?

1. In the first window, execute report **ZZSELECT_##** two times.
2. Switch to the second window and activate the SQL Trace.
3. Switch back to the first window and execute report **ZZSELECT_##** a third time, but now with trace running.
4. In the second window, stop the SQL trace and have a look at the trace file. Display the execution plan. Which access path was chosen? What are the estimated costs?
5. Estimate the cost of the index range scan by using the *Explain with hint* functionality.
6. Recompute the table statistics of table **ZLIPS_##** based on an 10% estimate.
7. Execute program **ZZSELECT_##** two times, now with the new table statistics. Then start the SQL Trace again and record the third execution of program **ZZSELECT_##**. Look at the execution plan and compare the cost of index range scan and full table scan.
8. Recompute the table statistics of table **ZLIPS_##** based on an 1% estimate.

Continued on next page

9. Recreate the secondary index ZLIPS_## and determine the estimated costs when using this index.
10. Delete the database statistics of table ZLIPS_##.

Result

You compiled a bunch of data for the access of table **ZLIPS_##** by program **ZZSELECT_##**. The estimated cost of the full table scan does not depend on the accuracy of the statistics but is determined by the number of blocks, that is, the table size. The estimated cost of an index range scan, however, strongly depends on the accuracy of the database statistics. Even with no statistics data available, the cost-based optimizer reports an estimated cost for the index range scan. The estimate, then, is based on misleading assumptions made by the optimizer. Therefore, an incorrect, much-too-low cost for the index range scan is evaluated. Finally, the index range scan is favored over the much faster full table scan.

Solution 22: The Cost-Based Optimizer in Action

Task:

Investigate an expensive SQL statement of your choice. Try to rationalize the access paths chosen by the cost-based optimizer. Are there dependencies of the computed costs on the table statistics?

1. In the first window, execute report **ZZSELECT_##** two times.
 - a) Call transaction SA38 and start report **ZZSELECT_##**.
2. Switch to the second window and activate the SQL Trace.
 - a) Start transaction ST05 and activate the trace.
3. Switch back to the first window and execute report **ZZSELECT_##** a third time, but now with trace running.
 - a) Call transaction SA38 and start report **ZZSELECT_##**.
4. In the second window, stop the SQL trace and have a look at the trace file. Display the execution plan. Which access path was chosen? What are the estimated costs?
 - a) Go to the second window and choose *Deactivate Trace*. Then choose *Display Trace*.
 - b) In the dialog box, choose *Display Trace List*.
 - c) Select an SQL statement and choose *Explain*. The execution plan is displayed in a tree-like presentation. The cost based optimizer has chosen a full table scan. Read the execution plan from bottom to the top, if necessary. Use the following table for subsequent tasks.

Statistics	Cost of		
	index range scan with table access by index Row ID (primary index)	full table scan	index range scan with table access by index Row ID (ZLIPS_##~M)
Compute			
Estimate P10			
Estimate P1			

Continued on next page

5. Estimate the cost of the index range scan by using the *Explain with hint* functionality.
 - a) You are still in the SQL Explain window of transaction ST05. Choose *Explain with hint*.
 - b) In the following dialog box, enter **INDEX (ZLIPS_##)**.
 - c) Press *Continue* and the estimated costs of the best index range scan is displayed.
6. Recompute the table statistics of table ZLIPS_## based on an 10% estimate.
 - a) Call transaction DB20.
 - b) Enter **ZLIPS_##** in the *Table* field. Correspondingly, enter **E** and **P10** in the *New Method* and *New Sample Size* fields.
 - c) Choose *Create statistics*.
 - d) Confirm with *Yes* in the following dialog box. You will see the message **Operation started** in the status line of your SAP GUI window.
 - e) After a while, choose *Update Info*. You should see a current day and time for the last refresh.
7. Execute program ZZSELECT_## two times, now with the new table statistics. Then start the SQL Trace again and record the third execution of program ZZSELECT_##. Look at the execution plan and compare the cost of index range scan and full table scan.
 - a) Follow the same procedure as in exercise steps 1 to 5.
8. Recompute the table statistics of table ZLIPS_## based on an 1% estimate.
 - a) Call transaction DB20.
 - b) Enter **ZLIPS_##** in the *Table* field.
 - c) Enter **E** and **P1** in the *New Method* and *New Sample Size* fields.
 - d) Choose *Create statistics*.
 - e) Confirm with *Yes* in the following dialog box. You see the message **Operation started** in the status line of your SAP GUI window.
 - f) After a while, choose *Update Info*. You should see a current day and time for the last refresh.

Continued on next page

9. Recreate the secondary index ZLIPS_## and determine the estimated costs when using this index.
 - a) Use transaction SE11 to create a new index, ZLIPS_##~M, for table ZLIPS_##.
 - b) Recompute the table statistics of table ZLIPS_## with exact calculation of the statistics.

Execute program ZZSELECT_## two times, now with the new table statistics. Then start the SQL trace again and record the third execution of program ZZSELECT_##. Look at the execution plan.
 - c) Recompute the table statistics of table ZLIPS_## based on a 10% estimate.

Execute program ZZSELECT_## two times, now with the new table statistics. Then start the SQL trace again and record the third execution of program ZZSELECT_##. Look at the execution plan.
 - d) Recompute the table statistics of table ZLIPS_## based on a 1% estimate.

Execute program ZZSELECT_## two times, now with the new table statistics. Then start the SQL trace again and record the third execution of program ZZSELECT_##. Look at the execution plan.
10. Delete the database statistics of table ZLIPS_##.
 - a) Call transaction DB20.
 - b) Enter **ZLIPS_##** in the *Table* field.
 - c) Choose *Delete statistics*. When prompted to confirm the deletion, choose *Yes*.
 - d) After a while, choose *Update Info*. In the status area, you should see a red traffic light and the message **Statistics missing**.

Result

You compiled a bunch of data for the access of table **ZLIPS_##** by program **ZZSELECT_##**. The estimated cost of the full table scan does not depend on the accuracy of the statistics but is determined by the number of blocks, that is, the table size. The estimated cost of an index range scan, however, strongly depends on the accuracy of the database statistics. Even with no statistics data available, the cost-based optimizer reports an estimated cost for the index range scan. The estimate, then, is based on misleading assumptions made by the optimizer. Therefore, an incorrect, much-too-low cost for the index range scan is evaluated. Finally, the index range scan is favored over the much faster full table scan.



Lesson Summary

You should now be able to:

- Describe the different factors that influence estimated cost



Unit Summary

You should now be able to:

- Name the different types of database statistics
- Describe the update statistics strategy recommended by SAP
- Detect problems with optimizer statistics
- Describe the different factors that influence estimated cost



Test Your Knowledge

1. The update statistics strategy recommended by SAP contains the following steps:

Choose the correct answer(s).

- ☐ A Watch and hide log entries
- ☐ B Monitor and evaluate table accesses
- ☐ C Scan and compile indexes
- ☐ D Check and update statistics
- ☐ E Write to file and delete old table entries

2. The main purpose of database statistics is:

Choose the correct answer(s).

- ☐ A To count read accesses
- ☐ B Its use by the cost-based optimizer
- ☐ C To filter data
- ☐ D To ease database backups

3. Which of the following parameters influence the behavior of the Oracle cost-based optimizer?

Choose the correct answer(s).

- ☐ A TUNE_OPTIMIZER
- ☐ B OPTIMIZER_MODE
- ☐ C CBO_ULTRA_WARP
- ☐ D DB_FILE_MULTIBLOCK_READ_COUNT
- ☐ E dbs/ora/use_hints



Answers

1. The update statistics strategy recommended by SAP contains the following steps:

Answer: D

The recommended update statistics strategy comprises two steps: check and update statistics.

2. The main purpose of database statistics is:

Answer: B

The cost-based optimizer uses the database statistics to evaluate access paths to tables. All other answers are incorrect.

3. Which of the following parameters influence the behavior of the Oracle cost-based optimizer?

Answer: B, D, E

OPTIMIZER_MODE should be set to CHOOSE, which activates the cost-based optimizer. DB_FILE... is important for parallel processing during full table or full index scans. db/ora/use_hints controls the use of generated Oracle hints.

Unit 11

Analyzing physical and logical layout

Unit Overview

This unit describes performance problems caused by the physical and logical layout of the system. These problems may originate from fragmented indexes and from I/O contention. Strategies to identify fragmented indexes and I/O contention are provided in this unit.



Unit Objectives

After completing this unit, you will be able to:

- Identify a fragmented index
- Identify I/O contention in the database

Unit Contents

Lesson: Fragmented Indexes	616
Exercise 23: Detecting and Defragmenting Fragmented Indexes	631
Lesson: I/O Contention.....	637

Lesson: Fragmented Indexes

Lesson Overview

This lesson explains how an index becomes fragmented and discusses possible strategies to identify fragmented indexes.



Lesson Objectives

After completing this lesson, you will be able to:

- Identify a fragmented index

Business Example

After data is archived, you detect performance problems in your system. These problems may be caused by fragmented indexes.

Index Fragmentation

If an SQL statement is expensive even when there is an appropriate index and the ABAP code is optimal, check whether the cause of the database time is index fragmentation. To understand fragmentation, consider the example of an index containing a million of records. If 99% of these records are deleted in the corresponding table and therefore in the index, the blocks that were occupied by the deleted records are not released, nor are the remaining valid entries grouped together. Now, not only is the major part of the index not serving any purpose, but the valid data is distributed across a large range of blocks. Therefore, an index range scan must read a large number of index blocks to return comparatively few data records. This condition is called fragmentation. To defragment the index, you need to reorganize it.

An index that is fragmented consists of empty blocks or branch and leaf pages with only a few valid entries. Fragmented indexes affect performance of the entire database, since data or index blocks of other tables are swapped out of the buffer by the newly read index blocks, which contain only a few records. In the figure below, nine index blocks and five data blocks are read in order to retrieve five rows of the table.

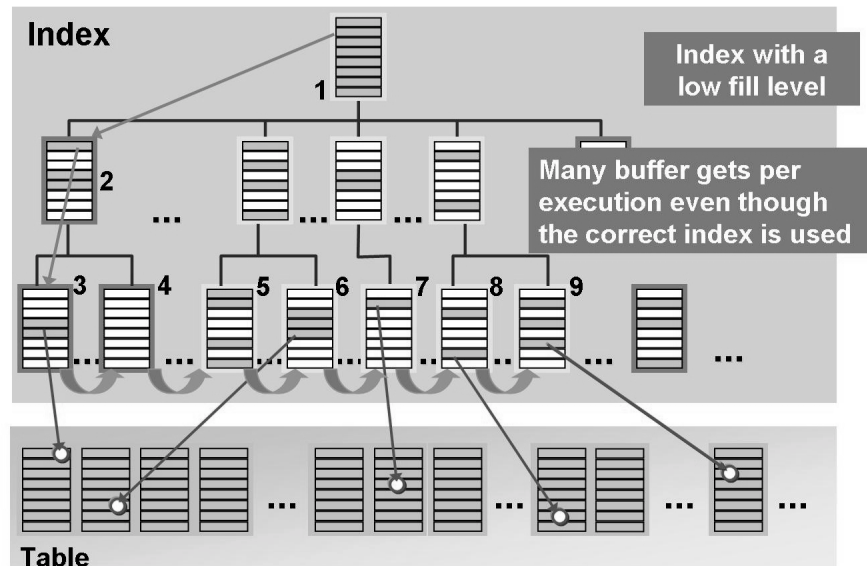


Figure 219: Fragmented Indexes

The fragmentation of an index specifies how well the space is used in the B* tree of an index. The smaller the space that is actually used, the more fragmented the index. In this case, the index may also be called unbalanced or degenerated.

In particular, index fragmentation affects tables that are subject to considerable data changes and that also index any column values that increase with time. RFC-table indexes are an example of indexes that typically may need reorganization.

Index fragmentation causes the following problems:

- Database growth

The greater the index fragmentation, the more space is unused in the index B* tree. This means that the index segment and the entire database grow more quickly than is necessary.

- Poor performance by the index range scan

If an index is fragmented, the index accesses using the index range scan may have to read more blocks than is actually necessary. The buffer gets and disk reads associated with this process increase the access time considerably.

- Poor global performance as a result of block displacements in the buffer pool

If, due to index fragmentation, more blocks must be read into the Oracle buffer pool than is necessary, blocks are unnecessarily displaced from other segments. This, in turn, means that the buffer must be repeatedly reloaded, which can impede the performance of the entire database.

Index fragmentation usually occurs when data is repeatedly deleted from the index and reinserted at a different location. In particular, this happens when a field is indexed whose values increase with time. If older entries are deleted, it is always from the left side of the index, while new entries are always inserted on the right side of the index. As a result, the index tree is more or less empty on the left side, but keeps growing on the right.

Indexes can become fragmented:



- After data has been archived
- After many records have been deleted
- In highly dynamic tables

Identifying a Fragmented Index

There are two approaches to measure index fragmentation:



- **Index storage quality**

Compares the available space with the used space.

The smaller the space actually used in comparison with the total available space, the lower the index storage quality and the more fragmented the index.

- **Deleted leaf rows**

Compares the number of deleted leaf rows with all leaf rows.

The larger the portion of the deleted leaf rows compared with all leaf rows, the more fragmented the index.

- **Comparing the index size and the table size**

If an index is larger than the table, this is an indicator that the index is fragmented.

Index Storage Quality

The storage quality compares the available space with the space used, and the two values are roughly determined as follows:

- The available space is determined by the number of index blocks used multiplied by the block size (8192 bytes), minus the administration overhead for block headers, and so on.
- The space used is determined by the number of index entries multiplied by the average length of an entry, plus six bytes for the Row ID (space used in the leaf blocks) and an overhead for the root and branch blocks.

The smaller the space actually used in comparison with the total available space, the lower the index storage quality and the more fragmented the index.

Keep in mind the following restrictions when determining the storage quality:

- Especially with small indexes, the space used is often very small compared with the available space (for example, a 20-byte entry in an 8K leaf block if the index only contains one row). Therefore, the storage quality is only useful to a certain extent with small indexes. This is not a severe restriction, since small indexes usually do not trigger performance problems due to fragmentation.
- It is generally difficult to correctly include **all** factors that contribute to the space calculation. For example, the mechanism of the index block splits, block-internal fragmentation due to record lengths, or storage parameters such as PCTFREE and PCTUSED also play important roles in determining how much space can potentially be used.

Deleted Leaf Rows

The number of deleted leaf rows compared with all leaf rows is another measure for the fragmentation of an index. The larger the portion of the deleted leaf rows compared with all leaf rows, the more fragmented the index.

This method of comparing the number of deleted leaf rows with all leaf rows has the following restrictions:

- If an entry is added in a leaf block, all deleted leaf rows are removed from this block. The block may then be almost empty. However, this is not recognized because deleted leaf rows no longer exist.
- The deleted leaf rows are only calculated in relation to the leaf blocks. The number of existing branch blocks is ignored. For example, a COALESCE always results in 0 deleted leaf rows even though the index still has the same number of branch blocks and, therefore, can still be fragmented.

Due to these reasons, both methods can only be an indicator of the level of index fragmentation but can never prove it. Therefore, fixed thresholds cannot be specified for these two methods to the point that we could say an index is fragmented in a performance-relevant way.

Although it is not possible to give a general recommendation for the percentage value index at which fragmentation becomes critical, the empirical values have the following tendencies:

- Storage quality of 50% or higher: No action required
- Storage quality between 25% and 50%: Action required in individual cases
- Storage quality of 25% or lower: The index should be rebuilt

It is logical to rebuild an index if it is clear that it occupies an unnecessarily large number of blocks in the buffer pool or that it is responsible for a large number of buffer gets as a result of the fragmentation.

Detecting Index Fragmentation

In the following section selected methods to detect index fragmentation are presented. You can find a comprehensive list of methods to detect index fragmentation in SAP Note 771929.

Estimation of the Storage Quality

To determine the index storage quality, SAP delivers report RSORATAD, which could be called directly in transaction SE38 or by using the DBA Cockpit (transaction DBACOCKPIT). Choose *Space* → *Segments* → *Detailed Analysis*. In the following window, enter the name of the index in the *Segment* column and press *Continue*. In the following list, select the index and double-click. Then select the register card *Storage* and choose *Storage Quality*.



DBA Cockpit: Space → Segments → Detailed Analysis

Owner	Segment	Partition	Type	Tablespace	Header File	Header Block	Size(MB)	Extents	Blocks	Init.Extent
SAPSR3	ZLIPS_00-M		INDEX	PSAPSR3	20	246.283	36,000	51	4.608	0,016

Main Data	Table + Indexes	Partitions	Extents	Storage	History
-----------	-----------------	------------	---------	----------------	---------

Main Data	Storage quality
-----------	------------------------

Field	Value
Space per block (byte):	8033
Space per index entry (byte):	27.99
Number of table rows:	1000000
Number of blocks (calc.):	4034.13
Number of blocks (alloc.):	4608
Number of blocks in freelists / groups:	0 / 0
Number of blocks (used):	4565
Index storage quality (%):	88.37

Figure 220: Analyzing Storage Quality

A percentage is specified for the storage quality. This percentage is a rough calculation, but generally provides a good overview of the storage quality. If the specified index storage quality percentage is high, you can assume that you do not need to reorganize the index. If a percentage less than 50% is specified here, a reorganization of the index is usually recommended. Unfortunately, the procedure is inaccurate for very small indexes and may display a small percentage even though the index was reorganized. You should therefore also check the *number of blocks* value. If this is lower than 10, we recommend that you use one of the other methods to find the storage quality.

Analyzing storage quality:



- Advantages:
 - A lock is not necessary.
- Disadvantages:
 - It can only be executed for one index at a time.
 - It has a long runtime.
 - It causes a high system load.
 - There is no support for bitmap indexes and partitioned indexes.
 - You can only trust the results up to a point because B*Tree blocks on the FREELIST are not included in the calculation.

For mass analysis and mass defragmentation of nonpartitioned B*Tree indexes, you can use report RSORAISQN. This report provides different heuristic algorithms for fast and exact analysis and provides different methods for defragmentation, such as coalesce or rebuild online. It is restartable and controlled stoppable. Different possibilities of parallelism and history of storage qualities are offered by this report. Furthermore, it gives you the ability to monitor currently running operations and to estimate further runtime.



REPORT: RSORAIQN

Index Storage Quality Analysis

Workingset Selection

ID

Object Selection

Connection Name

Index to

Table to

Tablespace to

Storage Quality (%) between and

Index size (kb) between and

Action Selection

☐ Show History

Workingset Start between and

☒ Only last Action of each Type

☒ Start Workingset

☐ Defragment

☐ Coalesce

☒ Rebuild online

Parallel degree ☒ Compute Statistics (Oracle 9)

☒ Check Locks ☒ Coalesce Tablespace

☒ Analyze

☐ Fast

☒ Exact

Parallel degree

☐ Show Status of Workingsets

☐ Increase Workingsetparallelity

☐ Stop Workingset

☐ Restart Workingset

☐ Cleanup Workingset

☐ Set Stop Status for Workingset

☐ Cleanup History

Days to keep

Figure 221: Index Storage Quality Analysis

To execute the report successfully, privileges need to be granted to the database user (SAPR3 or SAP<Schema-Id>) used locally or on the remote database. This can be done according to SAP Note 970538 with use of the attached SQL script `privilege_script..` This script includes the following privileges:

```

UNLIMITED TABLESPACE
CREATE TABLE
ALTER TABLESPACE
SELECT ON DBA_INDEXES
SELECT ON DBA_TABLES
SELECT ON DBA_SEGMENTS

```



Hint: To avoid problems that can occur when rebuilding/coalescing indexes please read the following SAP notes about “index rebuild online” and “index coalesce”: 682926, 869521, 904188, and 332677.

Mass defragmentation of indexes can cause massive archivelog generation and massive resource consumption (especially I/O).

Estimation of the Ratio of Leaf Rows and Deleted Leaf Rows

You can determine the leaf rows and deleted leaf rows of an index with the Oracle command `ANALYZE INDEX VALIDATE STRUCTURE`. This command can be executed using transaction DBACOCKPIT or on Oracle level.



Caution: This method locks parts of the index against updates, that is, we recommend that you only execute this option if there is a low user load on the system. Update statements on this index sometimes wait until the result of the storage analysis is returned.

In the SAP system, this method could be executed using the DBA Cockpit. Choose *Space* → *Segments* → *Detailed Analysis*. In the following window, enter the name of the index in the *Segment* column and press *Continue*. In the following list, select the index and double-click. Then select the register card *Main Data* and choose *Validate*.

This is how the current statistics are determined and displayed. In the *Analysis of B*-tree* section, the entries *entries* and *deleted* in the subsection *B*-tree leaf blocks* are relevant. The relationship *deleted/entries* should be less than 25%.



DBA Cockpit: Space → Segments → Detailed Analysis

Owner	Segment	P...	Type	Tablespace	Head...	Header Block	Size(MB)	Extents	Blocks	Init Extent	Next Extent	Min Extents	Max Extents
SAPSR3	ZLIPS_00-M		INDEX	PSAPSR3	20	246.283	36,000	51	4.608	0,016	0,000	1	2.147.483.645

Segments: Main data

Validate Coalesce Rebuild Last Analysis: 21.12.2007 11:06:53

General data	
Owner	SAPSR3
Segment Name	ZLIPS_00-M
Partition Name	
Type	INDEX
Tablespace	PSAPSR3
Header File	20
Header Block	246.283
Freelists	0
Freelist Group	
Relative FNO	20
Buffer Pool	DEFAULT

Sizes	
Size(Mb)	36,000
Extents	51
Blocks	4.608
Initial Extents(Mb)	0,016
Next Extent(Mb)	0,000
Min Extents	1
Max Extents	2.147.483.645
PCT Increase	0

Others	
Compressed	not calc.
NoLogging	not calc.

Figure 222: Ratio of Leaf Rows and Deleted Leaf Rows

When you execute the command `ANALYZE INDEX VALIDATE STRUCTURE` on Oracle level, the `INDEX_STATS` view is filled with the relevant statistics. However, these are only ever available for a single index in this view. As soon as another index is analyzed, the statistics of the first index are no longer available. Furthermore, these statistics are only available from this one session. If `INDEX_STATS` is selected from another session, no data record is returned.

As of Oracle 9i, the **online** addition can be attached to the `ANALYZE INDEX` command to minimize locking (`ANALYZE INDEX ONLINE`). Unfortunately, this online method does not create any statistics, which means that the addition is currently not useful for our purposes.

At Oracle level, this information can be collected for all indexes using a script. The same method as above applies here, but the storage quality for all indexes of the user `SAP<SCHEMA-ID>` can be checked with this script. Execute this only at times of little or no system activity, since some data records are locked during this check.

➡ **Note:** For further information, see SAP Note 444287.

Estimation of the ratio of leaf rows and deleted leaf rows:



- Advantages:
 - All indexes can be analyzed on Oracle level, since the estimation is script-based.
 - The estimation retrieves exact information about the index fragmentation.
- Disadvantages:
 - The table is locked for DML statements (`INSERT`, `UPDATE`, `DELETE`) during the analysis.
 - It has a long runtime.
 - It causes a high system load.

Index Size

Highly fragmented large indexes, in particular, can often be detected by comparing the table sizes. If an index is larger than the table, this is an indicator that the index is fragmented. It is rare that all of the most important table columns are contained in the same index.

Even if the index is smaller than the table, you may detect a possible fragmentation when you use a plausibility check based on the indexed columns and typical column entries. For example, if an index only indexes two columns with a maximum of five

characters each, while the entries in the table contain a large number of additional longer columns, the index is probably already fragmented even if it is only half the size of the table.

Comparing index size and table size:



- Advantages:
 - There is no significant system load.
 - The runtime is short.
 - A lock is not necessary.
- Disadvantages:
 - It is only a very rough indicator.
 - Blocks that were already allocated during an extent, but are not yet contained in the index tree, are included in the calculation even though they are irrelevant for the index fragmentation.

BRCONNECT Statistical Run (Oracle 9i and Lower)

Collecting statistics using the ANALYZE TABLE command provides additional values to calculate if an index is fragmented. Check the log files of the statistical runs of BRCONNECT for entries of the following type:

```
BR986W Index <index> is unbalanced - please rebuild the index
```

These entries indicate that the index is fragmented. Reasons for these entries might be:

- The index is unbalanced.
- The parameter STATS_CHANGE_THRESHOLD in init<DBSID>.sap is set to a very small value. The default is 50 (%).
- The precision of the index statistic calculation is too low, because the hard-coded method in BRCONNECT is overwritten by the method in DBSTATC with lower accuracy, or because the sample rows taken by Oracle are not representative enough. The index is therefore classified as unbalanced although it is not.

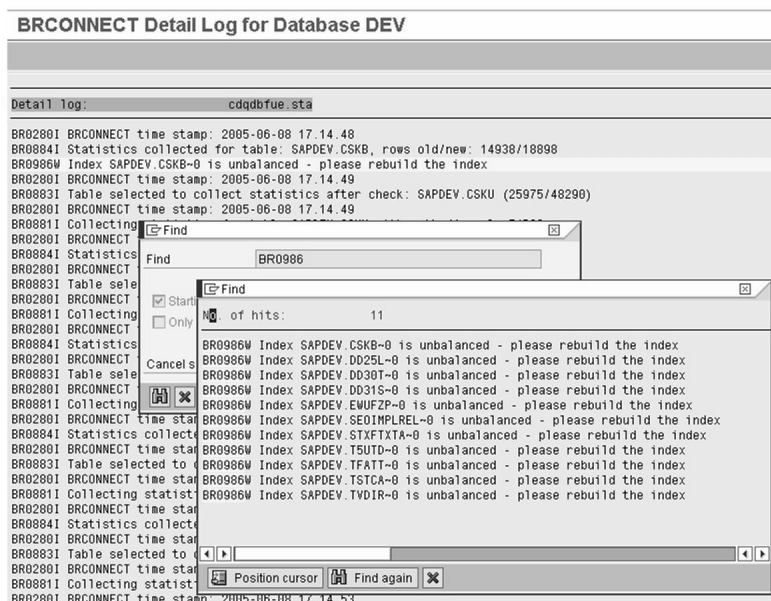


Figure 223: BRCONNECT Statistical Run (Oracle 9i and Lower)

To solve this problem, perform the following steps:

1. Check with the methods described earlier if the index is really unbalanced (see also SAP Note 444287). If so, recreate or rebuild the index. If the index is not unbalanced, proceed with the next step.
2. Check if `STATS_CHANGE_THRESHOLD` in `init<DBSID>.sap` is set to at least 20(%). If not, set it to at least 20%, or to the default of 50%. If `STATS_CHANGE_THRESHOLD` is okay, proceed with the next step.
3. Calculate statistics for this table with higher precision by using `DBSTATC` maintenance, or, if new statistics for the table are necessary, switch on table monitoring for this table so that the index is not considered for the decision. Increasing the precision increases the runtime of statistics calculation. If the runtime is unacceptably high, use table monitoring instead.



Note: For more details, see SAP Note 439783.

Analyzing index fragmentation using the results of BRCONNECT statistical runs (Oracle 9i and lower):



- Advantages:
 - A lock is not necessary.
 - There is no additional system load because it is a by-product of the statistics that are generated anyway.
- Disadvantages:
 - There is no guarantee that all fragmented indexes will be recognized or that the index mentioned in the warning is actually fragmented.

Index Defragmentation

Fragmented indexes reduce the performance of the database system. To improve data access, you can reorganize fragmented indexes using one of three options:



- DROP and CREATE index
- Rebuild index
- Coalesce index

DROP and CREATE Index

Using this method, you drop the index first and then create the index again. A disadvantage of this method is the long runtimes.

Rebuild Index

Advantages:

- Fast storage in a different tablespace
- Creates a new index tree
- Gives the option to change storage parameters without deleting the index
- As of Oracle 8i, you can avoid a lock on the table by specifying the ONLINE option. In this case, Oracle waits until the resource has been released, and then starts the rebuild. The **resource busy** error no longer occurs.

Disadvantages:

- Requires additional disk space and/or main memory

Coalesce Index

You can coalesce an index to deallocate internal free space. If space allocated to an index has never been used (that is, data has never been written to the data blocks), you can free such space for use by other objects in the tablespace. You can do this online without incurring downtime.

However, remember that the objects for which you have deallocated free space might themselves soon require new extents as they grow with inserts and updates. Therefore, a more permanent solution is to reorganize affected objects and also, if necessary, to extend the tablespace.

Advantages:

- Does not require any additional disk space
- The fastest solution
- Releases index leaf blocks for further usage
- Coalesces leaf blocks in the same branch of the index tree
- The table is not locked during the entire process

Disadvantages:

- Cannot move the index to another tablespace
- Cannot solve all problems of poor storage quality

SAP Tools to Defragment Indexes

You can use SAP tools to defragment an index:



- **BRSPACE**

With the BRSPACE function, you can rebuild a list of indexes using the REBUILD command:

```
brspace -f idrebuild -i <index_list>.
```

With the BRSPACE function, you can use the REBUILD command to rebuild a list of indexes.

- **Report RSANAORA**

Using report RSANAORA, you can rebuild an index with the REBUILD command.

- **DBA Cockpit**

In the SAP system, this method can be executed using the DBA Cockpit. Choose *Space* → *Segments* → *Detailed Analysis*. In the following window, enter the name of the index in the *Segment* column and press *Continue*. In the following list, select the index and double-click. Then select the register card *Main Data* and choose *Rebuild*.

- **Report RSORAISQN**

Report RSORAISQN allows you to rebuild indexes based on various criteria, such as index storage quality.



DBA Cockpit: Space → Segments → Detailed Analysis

DBA Cockpit interface showing the 'Segments' section. The top table lists segments with columns: Owner, Segment, P..., Type, Tablespace, Head..., Header Block, Size(MB), Extents, Blocks, Init.Extent, Next Extent, Min. Extents, and Max Extents. The selected segment is SAPSR3 ZLIPS_00~M, INDEX, PSAPSR3, with a size of 36,000 MB and 51 extents.

The 'Segments: Main data' section is displayed, showing the 'Rebuild' button. The 'Last Analysis' timestamp is 21.12.2007 11:06:53.

General data

Owner	SAPSR3
Segment Name	ZLIPS_00~M
Partition Name	
Type	INDEX
Tablespace	PSAPSR3
Header File	20
Header Block	246.283
Freelists	0
Freelist Group	
Relative FNO	20
Buffer Pool	DEFAULT

Sizes

Size(Mb)	36,000
Extents	51
Blocks	4.608
Initial Extents(Mb)	0,016
Next Extent(Mb)	0,000
Min Extents	1
Max Extents	2.147.483.645
PCT Increase	0

Others

Compressed	not calc.
NoLogging	not calc.

Figure 224: Rebuild Indexes using DBA Cockpit

Exercise 23: Detecting and Defragmenting Fragmented Indexes

Exercise Objectives

After completing this exercise, you will be able to:

- Use different tools to detect index fragmentation

Business Example

You detect performance problems in your system caused by index acces, even though you created suitable indexes. Now you have to check for fragmented indexes and, if necessary, defragmented the indexes.

Task 1:

Check if index **ZLIPS_##~M** is fragmented.

During this exercise the parameters that indicates fragmentation of the index are measured three times. Insert these values in the following table:

	Initial	After deletion of entries in ZLIPS_##~M	After index rebuild
Storage Quality			
deleted/entries			

1. Check the storage quality of index **ZLIPS_##~M** using transaction DBACOCKPIT or report RSORATAD.
2. Check the ratio of leaf rows and deleted leaf rows using transaction DBACOCKPIT.

Task 2:

Delete entries in table **ZLIPS_##** using report **ZZDEL_##**.

1. Execute your report, **ZZDEL_##**, which deletes nearly all entries of table **ZLIPS_##**.

Continued on next page

Task 3:

Repeat Task 1 to check if the fragmentation of index **ZLIPS_##~M** changed after you deleted table entries.

Task 4:

Rebuild index **ZLIPS_##~M**.

1. Rebuild the index **ZLIPS_##~M** using the “Alter Index Rebuild” function of the DBA Cockpit.

Task 5:

Repeat Task 1 to check if the fragmentation of index **ZLIPS_##~M** changed after you rebuilt the index.

Solution 23: Detecting and Defragmenting Fragmented Indexes

Task 1:

Check if index **ZLIPS_##~M** is fragmented.

During this exercise the parameters that indicates fragmentation of the index are measured three times. Insert these values in the following table:

	Initial	After deletion of entries in ZLIPS_##~M	After index rebuild
Storage Quality			
deleted/entries			

1. Check the storage quality of index **ZLIPS_##~M** using transaction DBACOCKPIT or report RSORATAD.
 - a) Switch to transaction DBACOCKPIT.
 - b) Choose *Space* → *Segments* → *Detailed Analysis*.
 - c) In the following window, enter the name of the index in the *Segment* column and press *Continue*.
 - d) In the following list, select the index and double-click.
 - e) Then select the register card *Storage* and choose *Storage Quality*.
The storage quality is displayed in the *Index storage quality* field.

Continued on next page

2. Check the ratio of leaf rows and deleted leaf rows using transaction DBACOCKPIT.
 - a) Switch to transaction “DBACOCKPIT”.
 - b) Choose *Space → Segments → Detailed Analysis*.
 - c) In the following window, enter the name of the index in the *Segment* and column press *Continue*.
 - d) In the following list, select the index and double-click. Then select the register card *Main Data* and choose *Validate*.

The number of leaf blocks (entries) and the number of deleted leaf rows (deleted) are displayed in the *B*-tree leaf blocks* subsection in the section *Analysis of B*-tree entries*. The deleted/entries relationship should be less than 25%.

Task 2:

Delete entries in table **ZLIPS_##** using report **ZZDEL_##**.

1. Execute your report, **ZZDEL_##**, which deletes nearly all entries of table **ZLIPS_##**.
 - a) Switch to transaction SE38 to execute report **ZZDEL_##**.

Task 3:

Repeat Task 1 to check if the fragmentation of index **ZLIPS_##~M** changed after you deleted table entries.

Task 4:

Rebuild index **ZLIPS_##~M**.

1. Rebuild the index **ZLIPS_##~M** using the “Alter Index Rebuild” function of the DBA Cockpit.
 - a) Switch to transaction DBACOCKPIT and choose *Space → Segments → Detailed Analysis*.
 - b) In the following window, enter the name of the index in the *Segment* column and press *Continue*.
 - c) In the following list, select the index and double-click. Then select the register card *Main Data* and choose *Rebuild*.

Continued on next page

Task 5:

Repeat Task 1 to check if the fragmentation of index **ZLIPS_##~M** changed after you rebuilt the index.



Lesson Summary

You should now be able to:

- Identify a fragmented index

Lesson: I/O Contention

Lesson Overview

This lesson describes how to identify I/O contention using the DBA Cockpit.



Lesson Objectives

After completing this lesson, you will be able to:

- Identify I/O contention in the database

Business Example

As an indicator of the performance of your system, you want to check if the I/O is evenly distributed across the disks of your filesystem. You would like to use different monitors in the SAP system to check the I/O contention.

Identifying I/O Contention in the Database

I/O contention refers to high I/O wait times for processes accessing the database. When numerous Oracle shadow processes and the database writer access the same disk at the same time, I/O contention is likely to occur.



I/O contention occurs when numerous shadow processes and the database writer access the same disk at the same time.

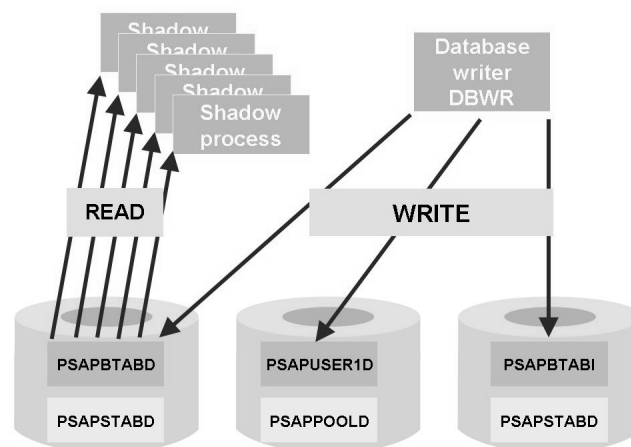


Figure 225: I/O Contention

I/O contention occurs if:

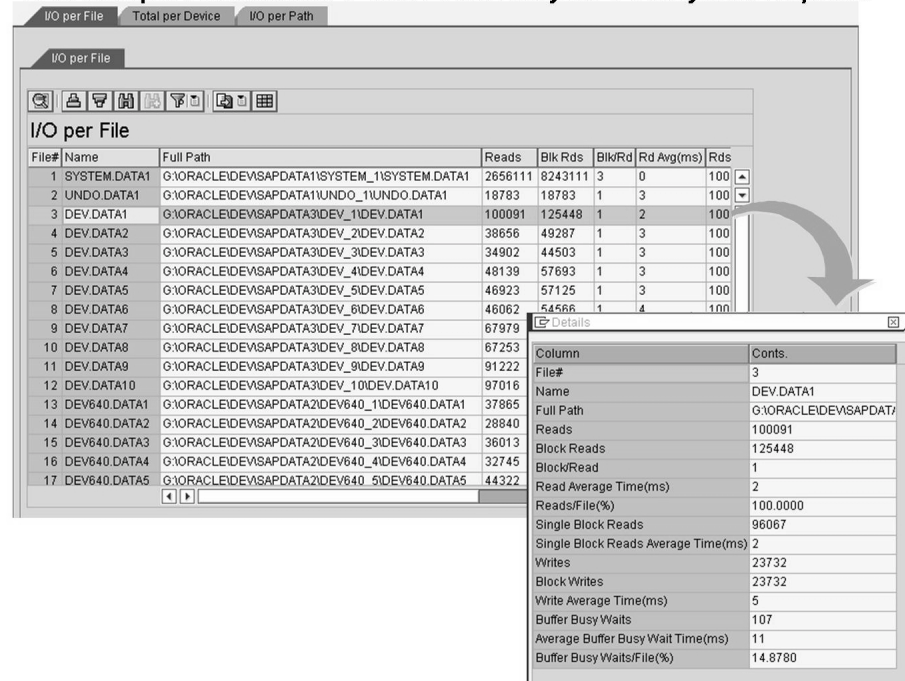


- The application design is inefficient due to expensive, unnecessary, or poorly qualified SQL statements.
- The I/O is not evenly distributed across many disks.
- Heavily accessed tables or indexes are not distributed or striped across many disks.
- The hardware configuration is incorrect (for example, many disks and not enough controllers).
- The filesystem configuration is incorrect.
- The OS configuration is incorrect.

I/O contention is often caused by application design problems, therefore, check this first.

To identify I/O contention in the database, use the **Filesystem requests** sub-monitor in the DBA Cockpit. This monitor helps you to minimize the time needed to read or write data from or to a file, so that you can identify the frequently used data files and put them on separate disks to avoid contention. Data file activity has an important effect on database performance.

To monitor filesystem requests, switch to the DBA Cockpit (transaction DBACOCKPIT) and choose *Performance* → *Wait Event Analysis* → *Filesystem Requests*.


DBA Cockpit: Performance → Wait Event Analysis → Filesystem Requests

Figure 226: Identifying I/O Contention in the Database

The **Filesystem Requests** monitor shows the following information based on the Oracle V\$FILESTAT view:

- **I/O per File**
Displays current statistics on physical file accesses per data file
- **Total per Device**
Displays current statistics on total physical file accesses per disk device; there are also entries for each file on the device
- **I/O per Path**
Displays current statistics about total physical file accesses per path; it represents the sum of accesses to the files



DBA Cockpit: Performance → Wait Event Analysis → Filesystem Requests

I/O per File										
Total per Device										
File#	Name / Device	Reads	Blk Rds	Blk/Rd	Rd Avg(ms)	Rds/File(%)	Sgl Blk Rds			
1	G:\ORACLE\DEVISAPDATA1\SYSTEM_1\SYSTEM.DAT	2656111	8243111	3	0	100.0000	1443870			
2	G:\ORACLE\DEVISAPDATA1\UNDO_1\UNDO.DAT	18783	18783	1	3	100.0000	18722			
23	G:\ORACLE\DEVISAPDATA1\DEVUSR_1\DEVUSR.DAT	921427	1114650	1	0	100.0000	890306			
== TOTAL FOR DEVICE G:\ORACLE\DEVISAPDATA1		3596321	9376544	5	1	100.0000	2352898			
13	G:\ORACLE\DEVISAPDATA2\DEV640_1\DEV640.DAT	37865	48215	1	1	100.0000	36290			
14	G:\ORACLE\DEVISAPDATA2\DEV640_2\DEV640.DAT	28840	55431	1	1	100.0000	24859			
15	G:\ORACLE\DEVISAPDATA2\DEV640_3\DEV640.DAT	36013	56870	1	1	100.0000	32898			
16	G:\ORACLE\DEVISAPDATA2\DEV640_4\DEV640.DAT	32745	54718	1	1	100.0000	29478			
17	G:\ORACLE\DEVISAPDATA2\DEV640_5\DEV640.DAT	44322	59849	1	1	100.0000	41987			
18	G:\ORACLE\DEVISAPDATA2\DEV640_6\DEV640.DAT	46639	62162	1	2	100.0000	44295			
19	G:\ORACLE\DEVISAPDATA2\DEV640_7\DEV640.DAT	27938	43485	1	1	100.0000	25593			
20	G:\ORACLE\DEVISAPDATA2\DEV640_8\DEV640.DAT	21151	31501	1	1	100.0000	19571			
21	G:\ORACLE\DEVISAPDATA2\DEV640_9\DEV640.DAT	19691	30027	1	1	100.0000	18121			
22	G:\ORACLE\DEVISAPDATA2\DEV640_10\DEV640.DAT	19503	29850	1	1	100.0000	17917			
== TOTAL FOR DEVICE G:\ORACLE\DEVISAPDATA2		314707	472108	10	1	100.0000	291009			
12	G:\ORACLE\DEVISAPDATA3\DEV_10\DEV.DAT	97016	114455	1	2	100.0000	94378			

I/O per Path										
I/O per Path										
Path	Reads	Blk Rds	Blk/Rd	Rd Avg(ms)	Rds/File(%)	Sgl Blk Rds	Sgl Blk Rds Avg(ms)	Writes	Blk Wrts	
G:\ORACLE\DEVISAPDATA1	3596321	9376544	5	1	100.0000	2352898	1	215053	238805	
G:\ORACLE\DEVISAPDATA2	314707	472108	10	1	100.0000	291009	2	1656	1656	
G:\ORACLE\DEVISAPDATA3	638243	768025	10	3	100.0000	617044	3	190447	190447	
G:\ORACLE\DEVISAPDATA4	3088	4046	1	5	100.0000	2321	6	149	192	

Figure 227: I/O Contention: Identify Tablespace and Data File

Check the *Rd Avg(ms)* (read average time) column for block reads, and the *Wrt Avg(ms)* (average time for writes) column for block writes. Use these values to identify “hot disks.” If the total number of reads and writes is relatively low, there is no I/O contention. If the total number of reads and writes is high, check if the average read time is higher than 20 ms. You can also identify hot disks by checking for values that deviate by more than 20% from the median value of the average read or write times.



Hint: Due to the various hardware configurations and disk speeds, the actual values can differ significantly from system to system. Contact your hardware vendor for specific information.

Because Oracle writes data blocks asynchronously, the average write time is not important. However, the average write time becomes important if Oracle is not able to handle the volume anymore. You can identify this situation by checking the following wait events: **write complete waits**, **free buffer waits**, and **log file switch** (checkpoint not complete).

The I/O per path monitor allows you to identify the sapdata mount points where I/O contention is occurring.

Solving the I/O contention Problem

To solve the problem of I/O contention, you can:



- Distribute the I/O evenly on the disks available.
- Distribute the I/O evenly on more disks (then would be necessary to hold the database files).
- Purchase faster disks.
- Move **hot spot** tables or indexes to their own tablespaces on their own disks (may be striped).

To identify the tablespace and the data file that has a bottleneck, you can break down the total I/O requests per filesystem by choosing *Total per device*. The tablespace and the data file can then be moved to another physical disk in the system.

Different hardware platforms may have bottlenecks in disk controller ports, motherboards, and back planes. Refer to your hardware vendor for I/O distribution guidelines.



Lesson Summary

You should now be able to:

- Identify I/O contention in the database



Unit Summary

You should now be able to:

- Identify a fragmented index
- Identify I/O contention in the database

Unit 12

Analyzing memory configuration

Unit Overview

This unit shows how you could monitor the Dynamic System Global Area and the Automatic Program Global Area. Monitoring the SGA focus is put on the utilization of the data buffer and the efficiency of the shared pool.



Unit Objectives

After completing this unit, you will be able to:

- Identify the data buffer hit ratio
- Monitor the size of the data buffer

Unit Contents

Lesson: Data Buffer Utilization	646
Exercise 24: Estimating Data Buffer Utilization.....	657

Lesson: Data Buffer Utilization

Lesson Overview

After activating dynamic System Global Area, it is important to monitor the memory usage of the SGA and the size of the data buffer. In this lesson, you will learn how to monitor the utilization of the data buffer.



Lesson Objectives

After completing this lesson, you will be able to:

- Identify the data buffer hit ratio
- Monitor the size of the data buffer

Business Example

After activating the Oracle dynamic System Global Area, you need to know how to monitor the SGA.

Monitoring the System Global Area

For the performance of your SAP system it is important to size the cache of the database appropriately for the needs of the application. Keep in mind that tuning the application's use of the caches can greatly reduce resource requirements. Therefore, it is important to monitor the usage of the database cache.

The main purpose of the System Global Area is to store data in memory for fast access. Therefore, the SGA should be within main memory. If pages of the SGA are swapped to disk, then the data is no longer quickly accessible. On most operating systems, the disadvantage of paging significantly outweighs the advantage of a large SGA, so you should ensure that the SGA fits into the main memory.

After an instance is started, you can display the SGA with the following command:

```
SQL>show sga
```

Example:



```
SQL>show sga
Total System Global Area 135339268 bytes
Fixed Size                453892 bytes
Variable Size            109051904 bytes
Database Buffers         25165824 bytes
Redo Buffers              667648 bytes
```

SQL>

The following areas of the SGA are displayed:

Total System Global Area

(Maximum) total size of the SGA (including all subareas) in bytes

Fixed Size

Contains general information on the status of the database and the instance; is required by the background processes

Does not contain any user data

Usually, this area is smaller than 100k

Variable Size

This area depends on the parameters `SHARED_POOL_SIZE`, `LARGE_POOL_SIZE`, and `JAVA_POOL_SIZE`.

Database Buffers

Size of the buffer cache; contains copies of the data blocks from the data files. The size of this area is defined by the parameter `DB_CACHE_SIZE` (or `BLOCK_SIZE * DB_BLOCK_BUFFERS`).

Redo Buffers

Circular memory in the SGA that contains the change information written into the redologs

Affected by the parameter `LOG_BUFFER`, but the displayed values *Redo Buffers* and the profile parameter `LOG_BUFFER` do not agree



Caution: Without using the dynamic SGA, the command `show sga` shows the amount of memory that was actually allocated by the SGA. With dynamic SGA, however, the system displays the maximum possible memory (based on `SGA_MAX_SIZE`), and not the memory that is actually allocated at the moment.



Hint: When sizing the SGA, make sure that there is enough memory for the PGA and any other SAP instance running on the system.

The size of the SGA can increase (for example, by enlarging the buffer cache) or shrink (for example, by reducing the buffer cache) due to an administrative command, or due to Oracle-internal, Oracle-controlled processes. The following Oracle performance views inform you about the status of SGA components and about size change operations:



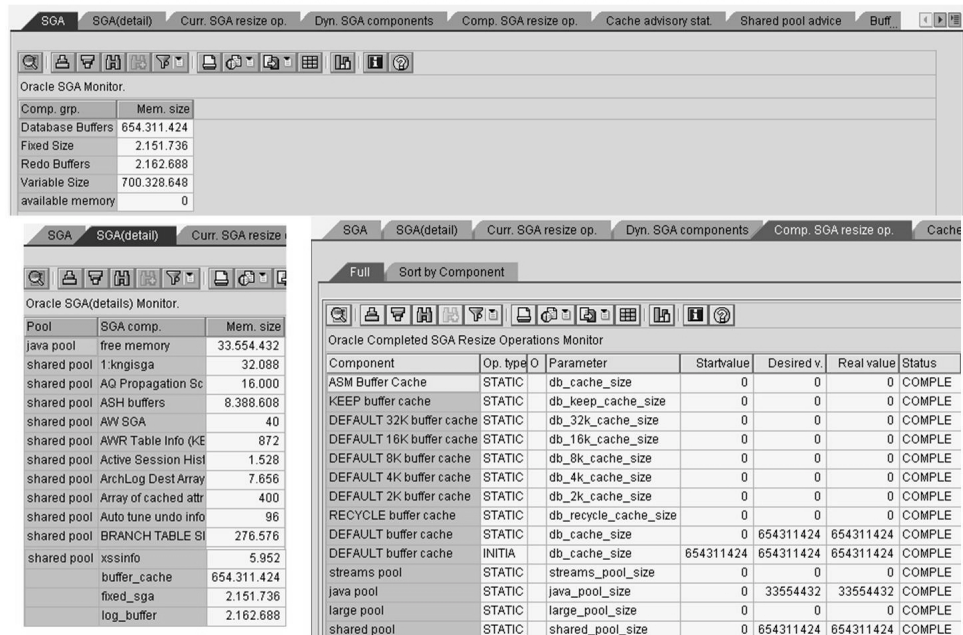
- **V\$SGA_CURRENT_RESIZE_OPS**
Current SGA size change operation(s)
- **V\$SGA_RESIZE_OPS**
List of the last 100 SGA size change operations carried out
- **V\$SGA_DYNAMIC_COMPONENTS**
Shows, for example, the current size and the previous minimum and maximum size of an SGA component, the number of size changes that have occurred on an SGA component, the type of the last size-changing operations on a component, and the current granule size.
- **V\$SGA_DYNAMIC_FREE_MEMORY**
Displays the memory available for future SGA enhancements

Monitoring the SGA Using the DBA Cockpit

In an SAP system, you can use the DBA Cockpit (transaction DBACOCKPIT) to monitor the information about the SGA provided by the Oracle performance views. The **SGA Monitor** sub-monitor is called as detailed analyses via *Performance* → *Statistical Information* → *SGA Monitor*. The SGA monitor displays the following main tabs:



- SGA
- SGA (detail)
- Curr. SGA resize ops.
- Dyn. SGA components
- Comp. SGA resize op.
- Cache advisory stat.
- Shared pool advice
- Buffer pool statistic


DBA Cockpit: Performance → Statistical Information → SGA Monitor

Figure 228: SGA Monitor

The following information is provided by the **SGA Monitor**:

- **SGA**

This screen provides basic information about SGA components. This information is equivalent to the result of the SQL*Plus command `show sga`.

The Oracle views `V$SGA` and `V$SGA_DYNAMIC_FREE_MEMORY` supply the information displayed. Size information is given in bytes.

- **SGA (detail)**

This tab page provides detailed information about SGA components.

The Oracle view `V$SGASTAT` supplies the information displayed.

- **Curr. SGA resize ops.**

The *Current SGA resize operations* screen shows information about currently ongoing resize operations. This view may be empty if no resize operations are being performed. In this case, a message will be issued stating that the related table is empty.

The information is derived from table `V$SGA_CURRENT_RESIZE_OPS`:

- Full

Groups the information by component and cumulate totals over all instances

- Sort by Component

Sorts the information by component; there are no instance totals

- **Dyn. SGA components**

The *Dynamic SGA components* screen displays information about dynamic SGA components. This view summarizes information based on all completed SGA resize operations since instance startup. All sizes are calculated in bytes.

The information is derived from table V\$SGA_DYNAMIC_COMPONENTS:

- Full

Groups the information by component and cumulate totals over all instances

- Sort by Component

Sorts the information by component; there are no instance totals

- **Comp. SGA resize op.**

The *Completed SGA resize operations* screen displays information about the last 100 completed SGA resize operations. This does not include in-progress operations. All sizes are calculated in bytes.

The information is derived from table V\$SGA_RESIZE_OPS:

- Full

Groups the information by component and cumulate totals over all instances

- Sort by Component

Sorts the information by component; there are no instance totals

- **Cache advisory stat.**

The *Cache advisory statistics* screen contains rows that predict the number of physical reads for the cache size corresponding to each row. The rows also compute a “physical read factor”, which is the ratio of the number of estimated reads to the number of reads actually performed by the real buffer cache during the measurement interval.

The information is derived from table V\$DB_CACHE_ADVICE. The cache size is given in MB.

- Full

Groups the information by component and cumulate totals over all instances; only shows size factor one

- Size for estimation

Values for size factor other than one are given

- **Shared pool advice**

This screen displays information about estimated parse time savings in the shared pool for different sizes. The sizes range from 50% to 200% of the current shared pool size, in equal intervals. The value of the interval depends on the current size of the shared pool.

The information is derived from table V\$SHARED_POOL_ADVICE. The cache size is given in MB.

- Full

Groups the information by component and cumulate totals over all instances; the data is filtered, and only size factor one is displayed

- Size for estimation

Values for size factor other than one are given

- **Buffer pool statistic**

This tab displays information about all buffer pools available for the instance. The “sets” pertain to the number of LRU latch sets.

The information is derived from table V\$BUFFER_POOL_STATISTICS. The block size is given in bytes.

Administration and Monitoring of the Data Buffer

Prior to tuning the buffer cache, make sure that your applications use the buffer cache effectively. SQL statements should be tuned to avoid unnecessary resource consumption. To ensure this, verify that frequently executed SQL statements that perform many buffer gets have been tuned.

The buffer cache activity should be examined to control whether the cache is well sized. Use the following statistics:

- Data buffer quality
- V\$DB_CACHE_ADVICE

Data Buffer Quality

The data buffer quality (or data buffer hit ratio) calculates how often a requested block is found in the buffer cache, without requiring disk access.



Rule of Thumb: Quality should be at least 94%

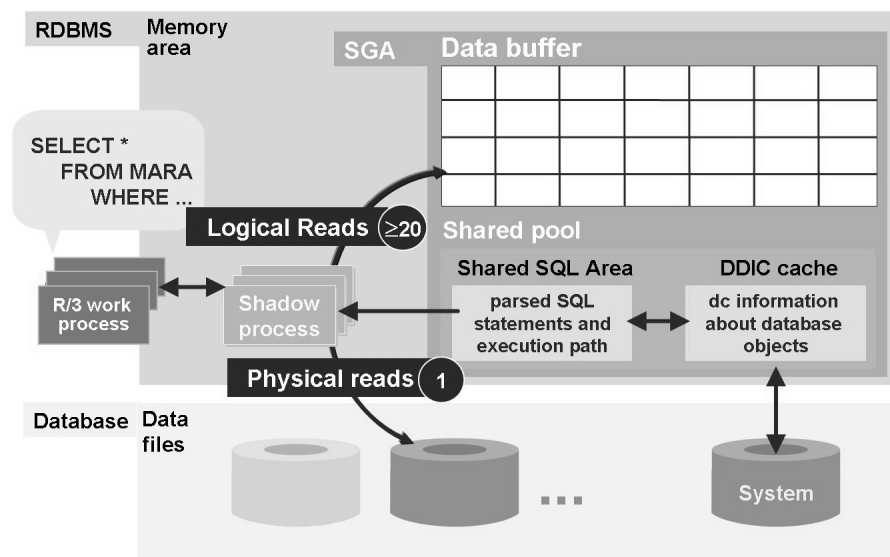


Figure 229: Data Buffer Quality

In the SAP system, you can monitor the data buffer quality with the the DBA Cockpit (transaction DBACOCKPIT). The data buffer quality is displayed in the **Performance Overview** monitor, which you can access by choosing *Performance* → *Performance Overview*. As a rule of thumb, the data buffer quality should be at least 94%.



1. Determining the ratio of physical reads and logical reads on the basis of V\$SYSSTAT:

$$\left(1 - \frac{\text{physical reads}}{\text{session logical reads}}\right) \times 100$$

2. Determining the ratio of physical reads and logical reads on the basis of V\$SYSSTAT, in which direct reads are not considered, since the data buffer is not read in direct reads:

$$\left(1 - \frac{\text{physical reads} - \text{physical reads direct} - \text{physical reads direct (lob)}}{\text{session logical reads} - \text{physical reads direct} - \text{physical reads direct (lob)}}\right) \times 100$$

3. Determining the ratio of physical reads and logical reads on the basis of V\$BUFFER_POOL_STATISTICS, in which direct reads are generally not considered in this view:

$$\left(1 - \frac{\text{physical reads}}{\text{DB_BLOCK_GETS} + \text{CONSISTENT_GETS}}\right) \times 100$$

Figure 230: Determination of Data Buffer Quality

There are various ways to determine the data buffer quality, as shown in the figure above.

Transaction ST04OLD uses method 1, while transactions DBACOCKPIT or ST04 use method 2. When there are many direct reads (parallel query, lobs, PSAPTEMP access, and so on), the hit ratio issued by DBACOCKPIT or ST04 may be better than the hit ratio calculated with the first method.

Method 3 delivers the same results as method 2, in which the hit ratio determined on the basis of V\$BUFFER_POOL_STATISTICS can be lower since, in many situations, less logical reads are logged than in V\$SYSSTAT.

The parameters used for calculating the data buffer quality are listed in the following table.

Parameters for Calculating the Data Buffer Quality

Parameter	Description
session logical reads	Total number of requests to access a block whether in memory or on disk
physical reads	Total number of requests to access a data block from disk
physical reads direct	Number of blocks read, bypassing the buffer cache, excluding direct reads for large objects
physical reads direct (lob)	Number of blocks read while reading large objects, bypassing the buffer cache

Interpreting the Results

The hit rate of the data buffer depends on many factors. A hit rate of 94% or higher does not necessarily mean that the data buffer is large enough. A good data buffer quality could wrongly indicate that the cache is adequately sized for the workload.

Consider the following situation: A few expensive SELECT statements are executed often. The data blocks of the accessed tables are “pinned” in the data buffer if the statement is executed often. Additionally, the number of buffer gets is very high because it is an expensive SELECT statement. Therefore, the hit rate for this statement is close to 100%. This type of statement increases the overall ratio of buffer gets to disk reads (the data buffer hit rate).

The data blocks accessed for such an expensive statement use a large fraction of the data buffer. Therefore, the size of the data buffer that can be used for all the other tables is significantly reduced. Disk accesses are performed to read data blocks for statements to the other tables, which could otherwise be read from the data buffer.

If the expensive SELECT statement is tuned, significantly fewer data blocks are read from the buffer. This can cause the hit rate to drop. However, the space available for all tables increases and the performance increases for all statements as there are fewer disk accesses).



- A data buffer hit rate of 94% or greater does not mean the buffer is large enough.
- Expensive SELECT statements can increase the data buffer hit rate because they significantly increase the number of successful buffer gets.
- If the ratio of buffer gets to user calls is much greater than 15, the data buffer hit rate is not a useful indicator.

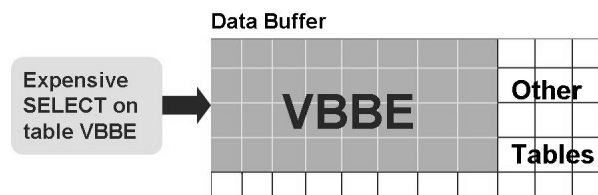


Figure 231: Interpreting Data Buffer Quality

The assessment of the data buffer hit rate is more reliable if you also consider the ratio of buffer gets to user calls. In a well-tuned system, this ratio is approximately 15. An expensive SELECT statement that increases the data buffer hit rate will increase this ratio as well. That is, if this ratio is much higher than 15, the data buffer hit rate cannot be used, and you must check for expensive SELECT statements.



Data Buffer			
Size (kB)	139,264	Logical reads	900,671,954
Quality (%)	98,5	Physical reads	13,348,677
		Physical writes	588,116
		Buffer busy waits	1,608
		Buffer wait time (s)	9
Shared pool			
Size (kB)	163,840	Log buffer	
DD-cache Quality (%)	97,5	Size (kB)	1,164
SQL area getratio(%)	97,1	Entries	6,368,269
SQL area pinratio(%)	99,8	Allocation retries	217
SQLAREloads/pins(%)	0,0169	Alloc fault rate(%)	0,0
		Redo log wait (s)	36
		Log files (in use)	8 (8)
Calls			
User calls	3,557,305	Recursive calls	1,123,947
User commits	73,827	Parse count	126,559
User rollbacks	116	User/recursive calls	3,2
		Log.Reads/User Calls	253,2

Quality: 98.5% looks extremely good
Reads / User calls: 253,2 is much higher than the recommended value of 15
 Therefore, the real quality is much less than 98.5%

Figure 232: Indication of Incorrect Data Buffer Quality

Using Cache Advisory Statistics

Using the performance view V\$DB_CACHE_ADVICE, you can set the size optimal of the SGA. Oracle collects the required data as soon as the Oracle parameter STATISTICS_LEVEL is set accordingly.

Collection of these statistics can be switched ON and OFF using the parameter STATISTICS_LEVEL, which can be changed dynamically. With this parameter, you can reduce overheads in CPU usage and memory. In order to collect these statistics, parameter STATISTICS_LEVEL should be set to TYPICAL. The STATISTICS_LEVEL parameter continues to activate more “advisor” Oracle servers internally. Parameter DB_CACHE_ADVICE, which was provided especially for V\$DB_CACHE_ADVICE, has been replaced by the parameter STATISTICS_LEVEL. DB_CACHE_ADVICE; it should therefore no longer be used.

The V\$DB_CACHE_ADVICE view shows the simulated miss rates for a range of potential buffer cache sizes. The predicted physical I/O activity is displayed for each simulated cache size.

V\$DB_CACHE_ADVICE compiles all data since the last time the instance was started. On live SAP systems with an average CPU utilization of 30% to 60% on the database server, the CPU overhead for the additional compiling and formatting of this information is acceptable.

In an SAP system, you can use the DBA Cockpit (transaction DBACOCKPIT) to monitor the information about the SGA provided by the Oracle performance views. The **SGA Monitor** sub-monitor is called as detailed analyses via *Performance* → *Statistical Information* → *SGA Monitor*.

The *Cache advisory statistics* screen contains rows that predict the number of physical reads for the cache size corresponding to each row. The rows also compute a **physical read factor**, which is the ratio of the number of estimated reads to the number of reads actually performed by the real buffer cache during the measurement interval. The data is filtered, and only size factor one is displayed. The information is derived from table V\$DB_CACHE_ADVICE. The cache size is given in MB.



SGA SGA(detail) Curr. SGA resize op. Dyn. SGA components Comp. SGA resize op. Cache advisory stat.								
Full Size for estimation								
Oracle DB Cache Advice Monitor								
Pool ID	Pool name	Block Size	Status	Cache(MB)	SizeFactor	CacheSize	Phys.R.F.	PhysReads
3	DEFAULT	8.192	ON	48	0,0769	5.949	1,5944	194.170.334
3	DEFAULT	8.192	ON	96	0,1538	11.898	1,4452	176.001.026
3	DEFAULT	8.192	ON	144	0,2308	17.847	1,2876	156.807.490
3	DEFAULT	8.192	ON	192	0,3077	23.796	1,2277	149.518.952
3	DEFAULT	8.192	ON	240	0,3846	29.745	1,0913	132.907.977
3	DEFAULT	8.192	ON	288	0,4615	35.694	1,0189	124.085.542
3	DEFAULT	8.192	ON	336	0,5385	41.643	1,0073	122.678.352
3	DEFAULT	8.192	ON	384	0,6154	47.592	1,0041	122.284.355
3	DEFAULT	8.192	ON	432	0,6923	53.541	1,0029	122.142.276
3	DEFAULT	8.192	ON	480	0,7692	59.490	1,0021	122.039.412
3	DEFAULT	8.192	ON	528	0,8462	65.439	1,0016	121.981.937
3	DEFAULT	8.192	ON	576	0,9231	71.388	1,0009	121.891.290
3	DEFAULT	8.192	ON	624	1,0000	77.337	1,0000	121.783.854
3	DEFAULT	8.192	ON	672	1,0769	83.286	0,9988	121.635.010
3	DEFAULT	8.192	ON	720	1,1538	89.235	0,9982	121.568.089
3	DEFAULT	8.192	ON	768	1,2308	95.184	0,9979	121.529.071
3	DEFAULT	8.192	ON	816	1,3077	101.133	0,9968	121.388.556
3	DEFAULT	8.192	ON	864	1,3846	107.082	0,9845	119.894.372
3	DEFAULT	8.192	ON	912	1,4615	113.031	0,9674	117.812.298
3	DEFAULT	8.192	ON	960	1,5385	118.980	0,9098	110.795.044

Reducing the size of the buffer cache

Current Size

Increasing the size of the buffer cache

Figure 233: Cache Advisory Statistics

The above figure shows that if the cache was 288 MB, rather than the current size of 624 MB, the estimated additional number of physical reads would not increase significantly. Increasing the cache size beyond 624 MB would not provide a significant benefit.

This view assists in cache sizing by providing information that predicts the number of physical reads for each potential cache size. The data also includes a physical read factor, which is a factor by which the current number of physical reads is estimated to change if the buffer cache is resized to a given value.

Exercise 24: Estimating Data Buffer Utilization

Exercise Objectives

After completing this exercise, you will be able to:

- Identify the data buffer hit ratio
- Monitor the size of the data buffer

Business Example

After activating the Oracle dynamic System Global Area, you need to know to monitor the SGA.

Task:

Determine the data buffer quality and make a prediction of a good size for the data buffer.

1. Find the data buffer quality of your system by using transaction DBACOCKPIT. To evaluate the significance of this value, use the ratio of logical reads to user calls.
2. Use the **Dynamic SGA components** monitor to evaluate information about all completed SGA resize operations.

This screen displays information about dynamic SGA components. This view summarizes information based on all completed SGA resize operations since instance startup. All sizes are calculated in bytes. The information is derived from table V\$SGA_DYNAMIC_COMPONENTS.

3. Use the **Cache advisory statistics** monitor to evaluate the optimal cache size for your system.

Solution 24: Estimating Data Buffer Utilization

Task:

Determine the data buffer quality and make a prediction of a good size for the data buffer.

1. Find the data buffer quality of your system by using transaction DBACOCKPIT. To evaluate the significance of this value, use the ratio of logical reads to user calls.
 - a) Switch to transaction DBACOCKPIT. The data buffer quality is displayed in the **Performance Overview** monitor via *Performance → Performance Overview*. The data buffer quality should be at least 94%.
 - b) Assessment of the data buffer quality is more reliable if you also consider the ratio of logical reads to user calls. In a well-tuned system, this ratio is approximately 15.
2. Use the **Dynamic SGA components** monitor to evaluate information about all completed SGA resize operations.

This screen displays information about dynamic SGA components. This view summarizes information based on all completed SGA resize operations since instance startup. All sizes are calculated in bytes. The information is derived from table V\$SGA_DYNAMIC_COMPONENTS.

- a) Switch to transaction DBACOCKPIT and choose *Performance → Statistical Information → SGA Monitor*.
 - b) Select *Dyn. SGA components*.
3. Use the **Cache advisory statistics** monitor to evaluate the optimal cache size for your system.
 - a) Switch to transaction DBACOCKPIT and choose *Performance → Statistical Information → SGA Monitor*.
 - b) Select *Cache advisory statistics* and choose *Size for estimation*.
 - c) Check for which cache size the number of physical reads would not significantly increase.



Lesson Summary

You should now be able to:

- Identify the data buffer hit ratio
- Monitor the size of the data buffer

Related Information

For more information about tuning SQL statements, see Unit 3, *Analyzing Application Design*.



Unit Summary

You should now be able to:

- Identify the data buffer hit ratio
- Monitor the size of the data buffer



Test Your Knowledge

1. Which V\$view can be used to determine whether the data buffer is well sized?
Choose the correct answer(s).
 - ☐ A V\$EVENT_NAME
 - ☐ B V\$PGA_TARGET_ADVICE
 - ☐ C V\$DB_CACHE_ADVICE
 - ☐ D V\$SGA_RESIZE_OPS



Answers

1. Which V\$view can be used to determine whether the data buffer is well sized?

Answer: C

Unit 13

Appendix: Analyzing memory configuration

Unit Overview



Unit Objectives

After completing this unit, you will be able to:

- Identify the efficiency of the shared pool
- Monitor the size of the shared pool
- Monitor the usage of the PGA

Unit Contents

Lesson: Efficiency of the Shared Pool	664
Exercise 25: Estimating the Efficiency of the Shared Pool	669
Lesson: Monitoring the Automatic Program Global Area.....	672
Exercise 26: Monitoring the Automatic PGA.....	681

Lesson: Efficiency of the Shared Pool

Lesson Overview

The shared pool size is an important factor for database performance. It should be just large enough to cache frequently accessed objects. In this lesson, you will learn how to monitor the efficiency of the shared pool.



Lesson Objectives

After completing this lesson, you will be able to:

- Identify the efficiency of the shared pool
- Monitor the size of the shared pool

Business Example

You have detected performance problems in your system, which are caused by an unsuitable size of the shared pool. You need to know how the efficiency of the shared pool can be monitored.

Monitoring the Usage of the Shared Pool

Shared pool size is an important factor in application performance. If the shared pool is too small, extra resources are used to manage the limited amount of available space. This consumes CPU and latching resources, and causes contention. Optimally, the shared pool should be just large enough to cache frequently accessed objects.

Identifying the Usage of the Shared Pool

The main components of the shared pool are:

- The shared SQL area, where parsed SQL statements are cached for shared access to all shadow processes
- The Data Dictionary cache (row cache), which holds the Oracle Data Dictionary information, including cost-based optimizer statistics



Hint: With the cost-based optimizer, the row cache will have substantially more information to hold than with the rule-based optimizer.



The shared pool caches parsed SQL statements and Data Dictionary information from the database.

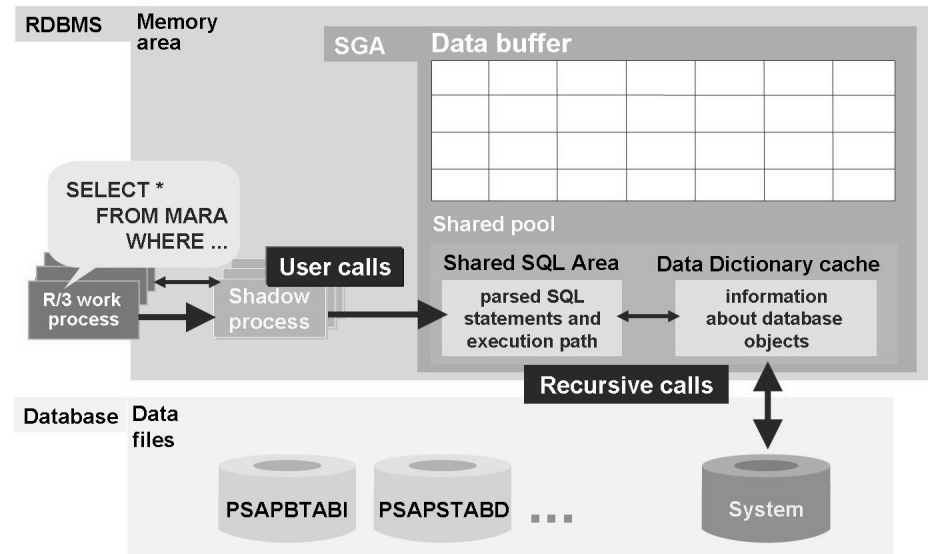


Figure 234: Identifying the Usage of the Shared Pool

The shared pool distinguishes between the following types of calls:

- **User calls**
A shadow process accessing the shared SQL Area for parsed SQL statements
- **Recursive call**
The Data Dictionary cache making a physical read to load Oracle Data Dictionary objects from the system tablespace

In the SAP system, you can monitor the efficiency of the shared pool by using the DBA Cockpit (transaction DBACOCKPIT). The respective parameters are shown in the **Performance Overview** monitor:

- **Shared SQL area**
 - The ratio of reloads to pins should be, at maximum, 0.04.
 - The pin ratio should be larger then or equal to 95%.
- **Dictionary cache**
 - The ratio of user calls to recursive calls should be at least 2 or higher.
 - The Data Dictionary cache quality should also be greater than 80%.

If these conditions are not fulfilled in normal operation, the value set for the shared pool may be too low. In this case, it may help to increase the Oracle parameter `SHARED_POOL_SIZE`.



Hint: The creation of new optimizer statistics is heavily based on recursive calls. You should therefore check whether the values meet the approximations in normal operation without such actions to make sure that creating statistics is not responsible for the values. To do this, you can use the history function in the Performance Overview. If so, the shared pool does not need to be enlarged.



Shared pool		Log buffer	
Size (kB)	163.840	Size (kB)	1.164
DD-cache Quality (%)	73,5	Entries	1.776.911
SQL area getratio(%)	91,4	Allocation retries	42
SQL area pinratio(%)	99,8	Alloc fault rate(%)	0,0
SQLA Reloads/pins(%)	0,0031	Redo log wait (s)	12
		Log files (in use)	8 (8)
Calls			
User calls	519.627	Recursive calls	170.254
User commits	1.066	Parse count	6.696
User rollbacks	21	User/recursive calls	3,1
		Log Reads/User Calls	8,1

Figure 235: Efficiency of the Usage of the Shared Pool

Using Shared Pool Advisory Statistics

The shared pool advisory statistics monitor the usage of the shared SQL area and predict how the shared SQL area will behave in shared pools of different sizes.

You can set the size of the SGA using the performance view `V$SHARED_POOL_ADVICE`. This view displays information about estimated parse time savings in different sizes of the shared pool. The sizes range from 50% to 200% of current shared pool size. The value of the interval depends on the current shared pool size.

In an SAP system, you can use the DBA Cockpit (transaction `DBACOCKPIT`) to monitor the information about the SGA provided by the Oracle performance views. The **SGA Monitor** sub-monitor is called as detailed analyses via *Performance* → *Statistical Information* → *SGA Monitor*.

The *Shared pool advice* screen contains the information predicted by `V$DB_CACHE_ADVICE`.

The *Parse Time* field refers to the amount of time saved by keeping library cache memory (shared SQL area) objects in the shared pool, as opposed to having to reload these objects.



Dyn. SGA components Comp. SGA resize op. Cache advisory stat. Shared pool advice							
Full Size for estimation							
Oracle Shared Pool Advice Monitor - Size For Estimation							
Size (MB)	Sizefactor	lib.cache	Mem.Obj.	ParseTime	ParseFact	CacheHits	
176	0,2821	69	5.606	369.092	0,9517	8.072.806	↑ Reducing the size of the shared pool
240	0,3846	131	9.872	386.110	0,9955	8.076.894	
304	0,4872	194	14.202	387.835	1,0000	8.078.137	
368	0,5897	257	16.825	387.837	1,0000	8.078.357	
432	0,6923	290	19.862	387.837	1,0000	8.078.400	
496	0,7949	290	19.862	387.837	1,0000	8.078.400	Current Size
560	0,8974	290	19.862	387.837	1,0000	8.078.400	
624	1,0000	290	19.862	387.837	1,0000	8.078.400	↓ Increasing the size of the Shared pool
688	1,1026	290	19.862	387.837	1,0000	8.078.400	
752	1,2051	290	19.862	387.837	1,0000	8.078.400	
816	1,3077	290	19.862	387.837	1,0000	8.078.400	
880	1,4103	290	19.862	387.837	1,0000	8.078.400	
944	1,5128	290	19.862	387.837	1,0000	8.078.400	
1.008	1,6154	290	19.862	387.837	1,0000	8.078.400	
1.072	1,7179	290	19.862	387.837	1,0000	8.078.400	
1.136	1,8205	290	19.862	387.837	1,0000	8.078.400	
1.200	1,9231	290	19.862	387.837	1,0000	8.078.400	
1.264	2,0256	290	19.862	387.837	1,0000	8.078.400	

Figure 236: Shared Pool Advisory Statistics



Hint: There are no specific Oracle parameters for increasing the size of the Dictionary cache or the size of the shared SQL area. By increasing the area for the entire shared pool, you also increase the amount of space available for both areas.

Exercise 25: Estimating the Efficiency of the Shared Pool

Exercise Objectives

After completing this exercise, you will be able to:

- Estimate the efficiency of the shared pool
- Monitor the size of the shared pool

Business Example

You have detected performance problems in your system, which are caused by an unsuitable shared pool size. You want to know how to monitor the efficiency of the shared pool.

Task:

Determine the usage of the shared pool.

1. Determine the quality of the shared SQL area and the Data Dictionary cache in your system using transaction DBACOCKPIT.
2. Use the shared pool advisory statistics to evaluate the optimal cache size for your system.

Solution 25: Estimating the Efficiency of the Shared Pool

Task:

Determine the usage of the shared pool.

1. Determine the quality of the shared SQL area and the Data Dictionary cache in your system using transaction DBACOCKPIT.
 - a) Switch to transaction DBACOCKPIT. The parameters you need to determine are displayed in the **Performance Overview** monitor. Choose *Performance* → *Performance Overview*.
 - b) The quality of the shared SQL area can be determined using the following parameters:
 - The ratio of reloads to pins (*SQLA.Reloads/pins*) should be, at maximum, 0.04.
 - The pin ratio (*SQL area pinratio*) should be larger than or equal to 95%.
 - c) The quality of the Data Dictionary cache can be determined using the following parameters:
 - The ratio of user calls to recursive calls (*User/recursive calls*) should be at least 2 to 1.
 - The Data Dictionary cache (*DD-cache quality*) quality should also be greater than 80%.
2. Use the shared pool advisory statistics to evaluate the optimal cache size for your system.
 - a) Switch to transaction DBACOCKPIT and choose *Performance* → *Statistical Information* → *SGA Monitor*.
 - b) Choose *Shared pool advice* → *Size for estimation*.
 - c) Check for which cache size the parse time would not increase significantly.



Lesson Summary

You should now be able to:

- Identify the efficiency of the shared pool
- Monitor the size of the shared pool

Lesson: Monitoring the Automatic Program Global Area

Lesson Overview

After activating automatic Program Global Area, it is important to monitor the memory usage of the PGA. This lesson explains how to monitor the PGA.



Lesson Objectives

After completing this lesson, you will be able to:

- Monitor the usage of the PGA

Business Example

After activating the Oracle automatic Program Global Area, you need to know how to monitor the PGA.

Monitoring

The size of the Program Global Area (PGA) can be monitored and tuned. Generally, performance can be improved by increasing the size of the PGA. The PGA should be at least big enough that no multipass sorts occur.

Even if the complete “Tuning PGA Automatic” process passes off automatically, it is nevertheless important and necessary to monitor this process to see whether it is suitably set up for the system environment or whether it still needs to be changed.

The following Oracle performance views contain information about PGA monitoring if automatic PGA is active:

- **V\$PGASTAT**
This view gives instance-level statistics on the PGA memory usage.
- **V\$SQL_WORKAREA**
This displays the execution statistics for work areas that are deallocated in case of cursors whose execution plan uses one or more work areas.
- **V\$SQL_WORKAREA_HISTOGRAM**
This view shows the number of work areas executed with optimal memory size, one-pass memory size, and multipass memory size since instance startup.
- **V\$SQL_WORKAREA_ACTIVE**
This view displays the work areas that are active in the instance. This view could be used to monitor the size of all active work areas and to determine if these active work areas spill over to a temporary segment.
- **V\$PROCESS**
This displays the PGA memory usage of Oracle processes. It contains the following new columns for monitoring automatic PGA: PGA_USED_MEM, PGA_ALLOC_MEM, and PGA_MAX_MEM.

The following views are relevant for the actual tuning:

- **V\$PGA_TARGET_ADVICE**
This view offers advice on what would happen if the PGA were increased or decreased in size by looking at PGA statistics since the instance was last started.

This view predicts how the cache hit percentage and over-allocation count statistics would be influenced if the size of the PGA were changed.
- **V\$PGA_TARGET_ADVICE_HISTOGRAM**
This view predicts how the number of work areas executed with optimal memory size, one-pass memory size, and multipass memory size statistics, which are displayed by the V\$SQL_WORKAREA_HISTOGRAM view, would be impacted if the value of the PGA_AGGREGATE_TARGET parameter were changed.

V\$SYSSTAT and V\$SESSTAT also contain new work area statistics for monitoring automatic PGA memory tuning. Before tuning the PGA memory, you should familiarize yourself with the statistics provided by the views mentioned above.

The contents of these views are valid from start of the instance, and are therefore initialized when restarting.

We recommend that you monitor these two views again after each change to parameter `PGA_AGGREGATE_TARGET`.

Monitoring the PGA uUsing the DBA Cockpit

You can use the DBA Cockpit (transaction `DBACOCKPIT`) to monitor the PGA. The **PGA Monitor** sub-monitor is called as detailed analyses via *Performance* → *Statistical Information* → *PGA Monitor*. It shows three main tabs:

- Status
- Snapshot
- PGA Advice



DBA Cockpit: Performance → Statistical Information → PGA Monitor

Status Snapshot PGA Advice		
PGA status SQL workarea SQL workarea histogra Workarea executions		
Oracle PGA Monitor		
Name	Statistic value	Unit
PGA memory freed back to OS	57.939.460.096	bytes
aggregate PGA auto target	509.912.064	bytes
aggregate PGA target parameter	629.145.600	bytes
bytes processed	75.263.399.936	bytes
cache hit percentage	92	percent
extra bytes read/written	5.889.986.560	bytes
global memory bound	104.857.600	bytes
max processes count	53	
maximum PGA allocated	269.248.512	bytes
maximum PGA used for auto workareas	140.804.096	bytes
maximum PGA used for manual workareas	0	bytes
memory used for workareas not in automatic management mode	125.709.312	bytes
over allocation count	0	
process count	46	
recompute count (total)	662.506	
total PGA allocated	125.709.312	bytes
total PGA inuse	62.530.560	bytes
total PGA used for auto workareas	0	bytes
total PGA used for manual workareas	0	bytes
total freeable PGA memory	29.425.664	bytes

Figure 237: PGA Monitor

PGA Monitor

Status

Status shows information about the PGA configuration. This monitor contains the following sub-monitors:

- PGA Status

This screen shows information about the PGA configuration based on the V\$PGASTAT view.

The following parameters are of interest here:

aggregate PGA target parameter

This parameter displays the value to which PGA_AGGREGATE_TARGET is set.

aggregate PGA auto target

This parameter displays the value that Oracle can use for work areas that run in automatic PGA mode. It is better if this value is not too small in relation to the total size of the PGA_AGGREGATE_TARGET aggregate PGA target parameter. There should still be sufficient space remaining for the work area.

over-allocation count

This value specifies how often the value of PGA_AGGREGATE_TARGET did not meet the minimum memory requests of the server process. As a result, Oracle had to add additional PGA memory internally again over and above the maximum value specified in PGA_AGGREGATE_TARGET. If PGA_AGGREGATE_TARGET is set optimally, the value of **over allocation count** should be zero. If **over allocation count** is greater than zero, the value of PGA_AGGREGATE_TARGET is too low. You must then increase PGA_AGGREGATE_TARGET.

cache hit percentage

This value shows the performance/efficiency of the Oracle automatic PGA memory management since the instance started. The ideal result of 100%, which is not usually achieved (nor does it have to be), means that the work areas of the server processes were the optimum size for all queries and that no one-pass or multi-pass operations occurred.

- SQL work area
 - View SQL WORKAREA

This tab shows information about the PGA configuration based on the V\$SQL_WORKAREA view. The size information is displayed in bytes, the times in seconds.

- Top 10 mem. cache con

This screen shows the top 10 consumers of memory cache, based on the V\$SQL_WORKAREA view. The information shown is the same as in the V\$SQL_WORKAREA view.

- One-multipass workarea

This table shows the work areas, the SQL text, the number of executions in the different modes, and the percentage of the total number of executions. The information shown is based on the V\$SQL and V\$SQL_WORKAREA views.

- SQL workarea histogram

- Histogram

This shows how many work areas were executed in optimal, one-pass, or multipass mode. The information is based on the V\$SQL_WORKAREA_HISTOGRAM view. From these statistics, which first provide an overview of the frequency and the memory requirement of SQL queries, you can draw conclusions as to whether the total PGA is sufficiently assessed.

- Percent optimal

This screen shows how many work areas were executed in optimal, one-pass, or multipass mode, and the percentage. The information is based on the V\$SQL_WORKAREA_HISTOGRAM view.

- Workarea executions

This tab page shows how often work areas were executed in different modes. The information shown is based on the V\$SYSSTAT view.

Snapshot

- Current Operations

This screen shows currently active operations with their work area memory allocation behavior. Depending on the application, the display in this view is very dynamic. The information is based on the V\$SQL_WORKAREA_ACTIVE view.

- PGA Memory Usage

This shows currently active operations. This screen displays OS process, the instance, memory information, and, if possible, the SQL text. The information is based on the V\$PROCESS view. The memory information is given in bytes.

PGA Advice

- Target Advice Size

This tab page predicts how the cache hit percentage and over-allocation count statistics displayed by the V\$PGASTAT performance view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter were changed. The prediction is performed for various values of the PGA_AGGREGATE_TARGET parameter, selected around its current value. The advice statistic is generated by simulating the past workload run by the instance. The information is based on the V\$PGA_TARGET_ADVICE view. The target advice size is given in MB.

- Advice Histogram

This sub-monitor predicts how statistics displayed by the V\$SQL_WORKAREA_HISTOGRAM dynamic view would be impacted if the value of the PGA_AGGREGATE_TARGET parameter were changed. This prediction is performed for various values of the PGA_AGGREGATE_TARGET parameter, selected around its current value. The advice statistic is generated by simulating the past workload run by the instance. It is possible to enter a target factor here. The target factor can be a number with two decimals. After entering the factor, press **ENTER** to execute the selection. The information is derived from V\$PGA_TARGET_ADVICE_HISTOGRAM. LOW_OPTIMAL_SIZE/low_kb and HIGH_OPTIMAL_SIZE/high_kb are given in KB.

Tuning the PGA Using Advisory Statistics

The aim of PGA tuning is to determine the optimum value for `PGA_AGGREGATE_TARGET`. The following conditions, which can be monitored in the **PGA Monitor**, should be fulfilled:

- Avoid PGA memory over-allocation
Set `PGA_AGGREGATE_TARGET` at least large enough so that the estimated over-allocation count = 0. If you set `PGA_AGGREGATE_TARGET` smaller (that is, too small), Oracle will automatically over-allocate this minimum amount of memory.
- Maximize the PGA cache hit percentage
The optimum, ideal cache-hit-percentage value would be 100%; however, this is only achievable with a very high memory requirement. Here, you should assess the marginal benefit of each additional MB and, based on that, decide whether it justifies the memory usage. The optimum value for `PGA_AGGREGATE_TARGET` is the value at which a further increase would not result in any relevant increase of the PGA cache hit percentage. If the database server has sufficient physical main memory for this optimum PGA size, set `PGA_AGGREGATE_TARGET` to this value. Otherwise, set `PGA_AGGREGATE_TARGET` to a size so that there is still enough memory remaining for the SGA.



Hint: A cache hit ratio of 90% for BW systems and of 60% for OLTP should be sufficient.

In addition to these two conditions, which help to find a good value for `PGA_AGGREGATE_TARGET`, the Oracle server contains the following two performance views, which provide information that will help set this parameter:

- `V$PGA_TARGET_ADVICE`
- `V$PGA_TARGET_ADVICE_HISTOGRAM`

The contents of these views are valid from start of the instance, and are therefore initialized when restarting. We recommend that you monitor these two views again after each change to `PGA_AGGREGATE_TARGET`.

In an SAP system, you can use the DBA Cockpit (transaction `DBACOCKPIT`) to monitor the information about the SGA provided by the Oracle performance views. The **PGA Monitor** sub-monitor is called as detailed analyses via *Performance* → *Statistical Information* → *PGA Monitor*.

The *PGA Advice* → *Target advice size* screen contains the information predicted by the Oracle view V\$PGA_TARGET_ADVICE.



DBA Cockpit:

Performance → Statistical Information → PGA Monitor

Status Snapshot PGA Advice		
Target Advice Size		Advice Histogram
Oracle PGA Target Advice Size		
TARGET (MB)	Value	Overall Cn
75	66	0
150	97	0
300	97	0
450	97	0
600	97	0
720	97	0
840	97	0
960	97	0
1,080	97	0
1,200	97	0
1,800	97	0
2,400	97	0
3,600	97	0
4,800	97	0

Figure 238: Target Advice Size

The contents of the Oracle view V\$PGA_TARGET_ADVICE_HISTOGRAM is displayed on the *PGA Advice* → *Advice Histogram* screen.



DBA Cockpit:

Performance → Statistical Information → PGA Monitor

Status Snapshot PGA Advice						
Target Advice Size Advice Histogram						
PGA target factor: 1,00						
PGA_TARGET_FACTO	LOW_KB	HIGH_KB	Optimal Executions	Onepass Ex	Multipasse	
1	2.147.483.648	4.294.967.295	0	0	0	
1	1.073.741.824	2.147.483.648	0	0	0	
1	536.870.912	1.073.741.824	0	0	0	
1	268.435.456	536.870.912	0	0	0	
1	134.217.728	268.435.456	0	0	0	
1	67.108.864	134.217.728	0	0	0	
1	33.554.432	67.108.864	0	0	0	
1	16.777.216	33.554.432	0	0	0	
1	8.388.608	16.777.216	0	0	0	
1	4.194.304	8.388.608	0	0	0	
1	2.097.152	4.194.304	0	0	0	
1	1.048.576	2.097.152	0	0	0	
1	524.288	1.048.576	0	0	0	
1	262.144	524.288	0	0	0	
1	131.072	262.144	0	0	0	
1	65.536	131.072	27	21	0	
1	32.768	65.536	2	0	0	
1	16.384	32.768	77	0	0	
1	8.192	16.384	2.893	0	0	
1	4.096	8.192	2.549	0	0	
1	2.048	4.096	66	0	0	
1	1.024	2.048	2.846	0	0	
1	512	1.024	14.178	0	0	

Figure 239: Cache Advice Histogram

Exercise 26: Monitoring the Automatic PGA

Exercise Objectives

After completing this exercise, you will be able to:

- Monitor the usage of the PGA

Business Example

After activating the Oracle automatic Program Global Area, you need to know how to monitor the PGA.

Task:

Use the DBA Cockpit to monitor the usage of the PGA.

1. Estimate the actual size of the PGA and compare it with the Oracle parameter `PGA_AGGREGATE_TARGET`.
2. Check how often the value of `PGA_AGGREGATE_TARGET` did not meet the minimum memory requests of the server process. Use transaction `DBACOCKPIT`.
3. Check how many work areas were executed in optimal, one-pass, or multipass mode. Use transaction `DBACOCKPIT`.

Solution 26: Monitoring the Automatic PGA

Task:

Use the DBA Cockpit to monitor the usage of the PGA.

1. Estimate the actual size of the PGA and compare it with the Oracle parameter `PGA_AGGREGATE_TARGET`.
 - a) Switch to transaction `DBACOCKPIT` and choose *Performance* → *Statistical Information* → *PGA Monitor*.
 - b) Choose *Status* → *PGA status*.

The value of the Oracle parameter `PGA_AGGREGATE_TARGET` is displayed in the *aggregate PGA target parameter* field.

The actual size of the PGA is displayed in the *total PGA in use* field.
2. Check how often the value of `PGA_AGGREGATE_TARGET` did not meet the minimum memory requests of the server process. Use transaction `DBACOCKPIT`.
 - a) Switch to transaction `DBACOCKPIT` and choose *Performance* → *Statistical Information* → *PGA Monitor*.
 - b) Choose *Status* → *PGA status*. The *over allocation count* field displays how often Oracle had to add additional PGA memory over and above the maximum value specified in `PGA_AGGREGATE_TARGET`.

If the *over allocation count* is greater than zero, this means that the value of `PGA_AGGREGATE_TARGET` is too low. You must then increase `PGA_AGGREGATE_TARGET`. If `PGA_AGGREGATE_TARGET` is set optimally, the value should be zero.
3. Check how many work areas were executed in optimal, one-pass, or multipass mode. Use transaction `DBACOCKPIT`.
 - a) Switch to transaction `DBACOCKPIT` and choose *Performance* → *Statistical Information* → *PGA Monitor*.
 - b) Choose *Status* → *SQL workarea histogram* → *Percent optimal*.



Lesson Summary

You should now be able to:

- Monitor the usage of the PGA



Unit Summary

You should now be able to:

- Identify the efficiency of the shared pool
- Monitor the size of the shared pool
- Monitor the usage of the PGA



Test Your Knowledge

1. The main components of the shared pool are the _____ and the _____.

Fill in the blanks to complete the sentence.

2. The PGA should be at least large enough so that the estimated over-allocation count = 0.

Determine whether this statement is true or false.

- ☐ True
- ☐ False



Answers

1. The main components of the shared pool are the shared SQL area and the Data Dictionary cache.

Answer: shared SQL area, Data Dictionary cache

2. The PGA should be at least large enough so that the estimated over-allocation count = 0.

Answer: True

If the PGA were set to a smaller value, Oracle would over-allocate this minimum amount of memory.



Course Summary

You should now be able to:

- Describe the architecture of an Oracle database
- Use database management tools provided by SAP
- Define a backup strategy
- Perform backups with use of SAP database management tools
- Perform a restore/recovery of the database using SAP tools
- Explain the concepts of Oracle data storage management
- Use SAP tools to run regular database checks
- Solve storage-related problems
- Define a strategy for refreshing the statistics used by the cost-based optimizer
- Describe Oracle cache management
- Use the SAP database monitor to analyze performance problems
- Detect expensive SQL statements
- Describe how incorrect indexes affect performance
- Analyze memory configuration of the Oracle database
- Analyze physical and logical layout of the database

Index

A

automatic Program Global Area, 390

B

B* tree index, 536
bitmap index, 537
busy wait time, 439

C

cache advisory statistics, 655
cost-based optimizer, 513

D

data buffer quality, 651
database, 365
database buffer gets, 498
database disk reads, 498
Database Monitor, 494
database physical reads, 498
database reads, 498
database transaction, 516
dynamic System Global Area, 371

E

exclusive lock waits, 522
expensive select, 483
expensive SQL statement, 483
expensive statement, 483
Explain
 analyze function, 514
 estimated costs, 513
 table/index statistics, 514

F

filesystem requests, 638

full table scan, 548

I

I/O contention, 637
idle wait time, 439
index
 coalesce, 628
 creation, 553
 filter, 543
 fragmentation, 616
 rebuild, 627
index fragmentation, 616
index range scan, 541
index skip scan, 514, 545
index unique scan, 541
instance, 365

M

missing index, 561
multipass sort, 387

O

one-pass sort, 387
Oracle enqueue, 520
Oracle lock, 520
Oracle Session Monitor, 495
Oracle Wait Interface, 436

P

primary index, 539
Program Global Area, 369, 386
 monitoring, 672

R

recursive calls, 665
recursive statement, 500

S

- SAP transaction, 517
- secondary index, 540
- selectivity analysis, 554
- shared cursor cache, 499
- shared pool, 375
 - monitoring, 664
- shared pool advisory statistics, 666
- shared SQL area, 376
- SQL Trace, 508
- statistical records, 492
- storage quality, 620
- System Global Area, 369
 - monitoring, 646
- System Global Area (SGA), 366

T

- tablespace, 365
- trace list, 512
- transaction profile, 490

U

- unselective index range scan, 544
- user calls, 665

W

- wait event, 437
 - busy wait time, 439
 - idle wait time, 439
 - monitoring, 442
 - tuning, 446
- work process overview, 487
- workload analysis, 490

Feedback

SAP AG has made every effort in the preparation of this course to ensure the accuracy and completeness of the materials. If you have any corrections or suggestions for improvement, please record them in the appropriate place in the course evaluation.